



Documentation for Service Desk Server 3.2

Contents

JIRA applications overview	4
Permissions overview	7
Installing JIRA Service Desk	14
Getting started with JIRA Service Desk	14
Getting started for service desk admins	16
Setting up your service desk	17
Creating service desk request types	19
Making queues for your service desk teams	22
Adding service desk agents	23
Customize your service desk channels	25
Bring your service desk to the next level	27
Introduce customers to your service desk	29
Explore a sample project	31
Getting started for service desk agents	32
Administering service desk projects	34
Automating your service desk	34
Configuring the customer portal	37
Configuring service desk notifications	41
Managing access to your service desk	42
Receiving requests by email	43
Managing the email channel	46
Troubleshooting issues with the email channel	46
Setting up approvals	51
Setting up queues for your team	54
Setting up request types	56
Troubleshooting issues with request types	60
Workflows	61
Setting up service desk users	61
Managing project role memberships	66
Using JIRA applications with Confluence	66
Using JIRA applications with HipChat	69
Default service desk project configuration	72
Working on service desk projects	74
Using service desk queues	75
Working with issues	76
Adding request participants	77
Attaching files and screenshots to issues	78
Creating issues and sub-tasks	80
Creating issues using the CSV importer	82
Editing and collaborating on issues	87
Linking issues	91
Editing multiple issues at the same time	95
Scheduling an issue	100
Moving an issue	102
Visual editing	102
Customizing the issues in a project	103
Logging work on issues	104
Approving a service desk request	107
Searching for issues	108
Basic searching	112
Quick searching	114
Advanced searching	117
Advanced searching - fields reference	123
Advanced searching - keywords reference	151
Advanced searching - operators reference	154
Advanced searching - functions reference	163

Search syntax for text fields	179
Saving your search as a filter	185
Working with search results	189
Constructing cron expressions for a filter subscription	199
Configuring dashboards	200
Adding and customizing gadgets	203
Gadgets for JIRA applications	204
Managing your user profile	210
Allowing OAuth access	212
Requesting add-ons	215
Using keyboard shortcuts	215
Organizing work with components	217
Organizing work with versions	218
Raising requests on behalf of customers	221
Using JIRA on a mobile device	221
Setting up service desk reports	223
Setting up SLAs	224
Reporting on SLAs	230
Example: creating a basic SLA	232
Example: creating an SLA that doesn't track continuous time	233
Example: creating an SLA with multiple cycles	233
Example: creating SLAs based on due dates	234
Serving customers with a knowledge base	235
Using the help center	237
Collecting customer satisfaction (CSAT) feedback	239
JIRA Service Desk best practices	241
Best practices for designing the customer portal	242
Best practices for IT teams using JIRA Service Desk	244
Service request fulfillment	250
Change management	253
Problem management	262
Incident management	267
Automating the calculation of priority based on impact and urgency values on issue creation	272
Getting help with JIRA Service Desk	274

JIRA applications overview


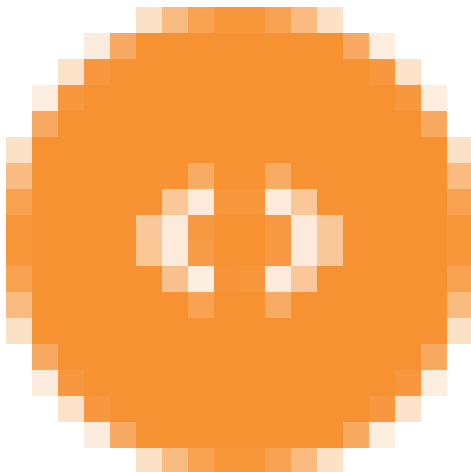
JIRA Service Desk licensing overview


The JIRA family of applications (JIRA Software, JIRA Service Desk, JIRA Core) are built on the JIRA platform and can be used in any combination on the same instance. Depending upon of your setup, users can be licensed to one, all, or any combination of these applications. Read on to understand how JIRA Service Desk licensing and roles affect what agents, customers, and other JIRA application users can do.

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.


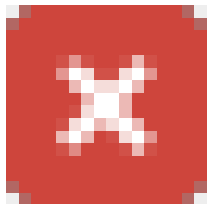
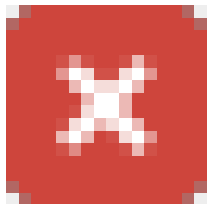
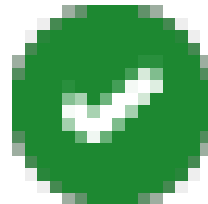
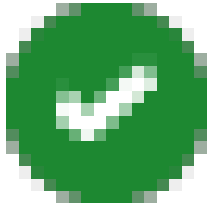
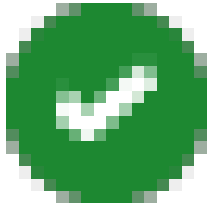
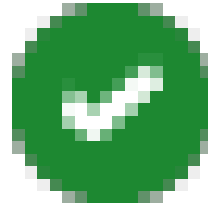
Application	Project type	Application specific feature set
JIRA Core	 Business projects	<ul style="list-style-type: none">• Available to all licensed users
JIRA Software	 Software projects	<ul style="list-style-type: none">• Integration with development tools• Agile boards• Release hub for software versions

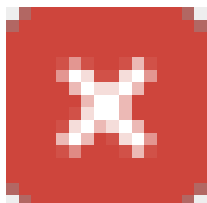

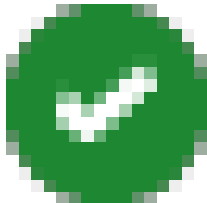
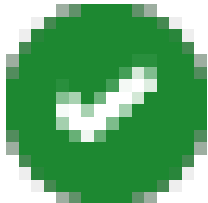


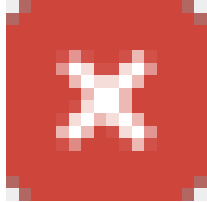

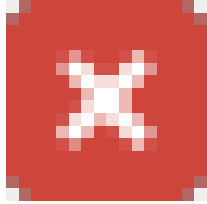
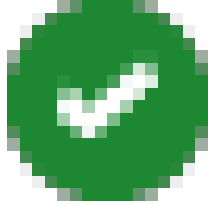
JIRA Service Desk	 <p>Service Desk projects</p>	<ul style="list-style-type: none"> • Service Level Agreements (SLAs) • A customizable web portal for customers • Permission schemes allowing customer access
-------------------	--	---

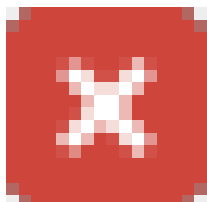
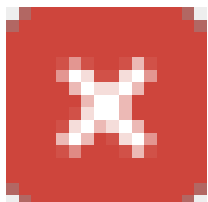

All users that can log in to a JIRA instance will be able to see all the projects in that instance (pending permissions), but they will only be able to see the application-specific features when they have application access. For example, a Software project is able to display information from linked development 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
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			
--	------------------------------------	------	---	--	---

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](#).

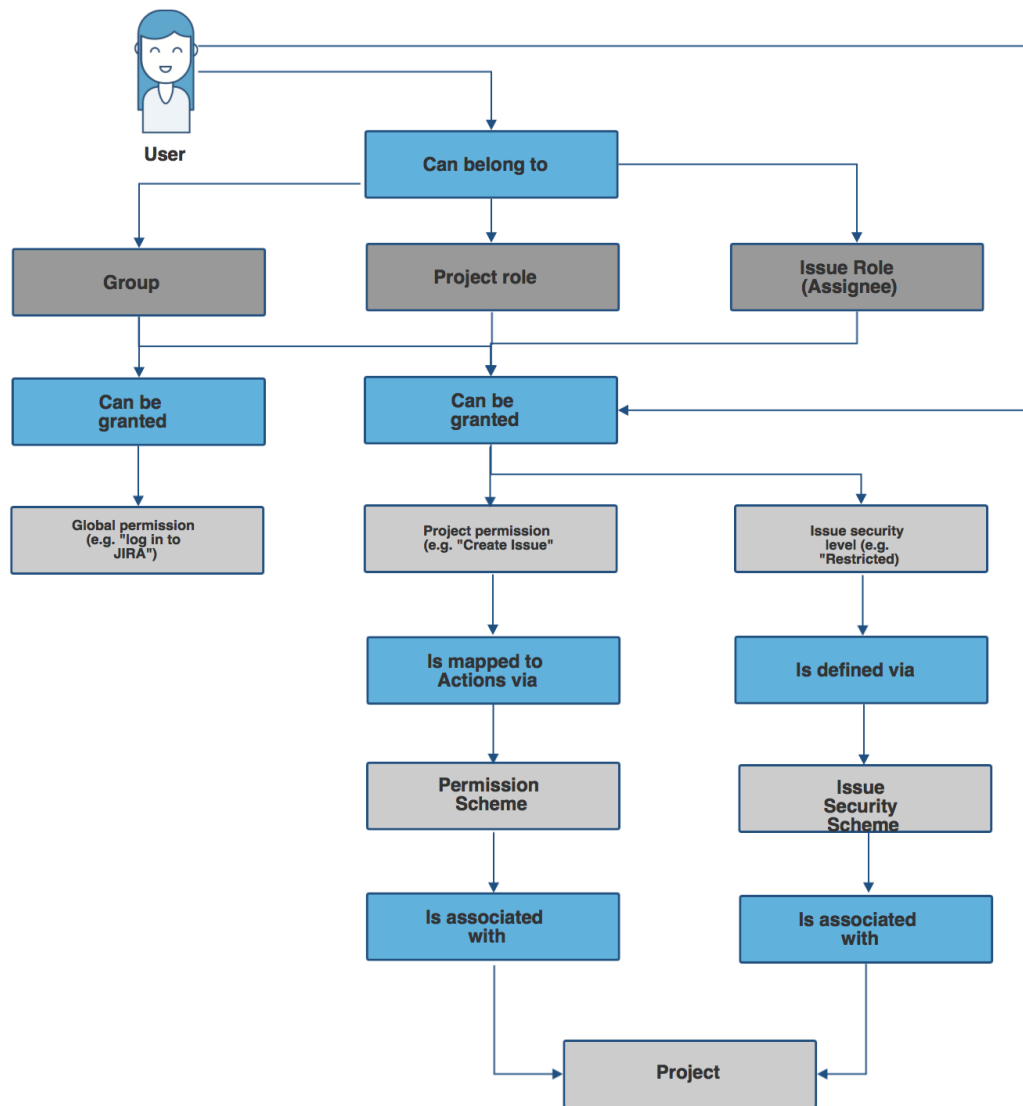
Types of permissions

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

JIRA Service Desk global and project permissions

JIRA Service Desk provides a standard permission scheme (JIRA Service Desk Permission scheme for *project*) that automatically gives your service desk users the correct permissions for the project role they are in. For example, adding agents to your service desk will add users to the Service Desk Team role. This role gives them access to JIRA Service Desk projects to which they're assigned and also allows them to work on issues.

Global permissions

At installation time, JIRA Service Desk creates a global permission named **JIRA Service Desk agent access**. If agent based pricing is enabled for the instance, users who require access to agent views or functionality need to have this permission. The number of users who are granted this permission determines how many agent licenses are used on the system.

Project permissions

This table shows the permission configuration for a standard service desk project permission scheme:

Project Permissions	Users / Groups / Project roles	Explanation
Administer Projects	Project Role (Administrators)	Permission to administer a project. This includes the ability to edit project role membership, project components, project versions and certain project details (Project Name, URL, Project Lead, Project Description).
Browse Projects	<ul style="list-style-type: none"> Service Desk Customer - Portal Access Project Role (Service Desk Team) Project Role (Administrators) 	Permission to browse projects, use the Issue Navigator and view individual issues (except issues that have been restricted via issue security). Many other permissions are dependent on this permission , e.g. the 'Work On Issues' permission is only effective for users who also have the 'Browse Projects' permission.
View Development Tools	<ul style="list-style-type: none"> Project Role (Administrators) 	
View (Read-Only) Workflow	<ul style="list-style-type: none"> Project Role (Service Desk Team) Project Role (Administrators) 	Permission to view the project's 'read-only' workflow when viewing an issue. This permission provides the 'View Workflow' link against the Status field of the ' View Issue ' page.

Issue Permissions	Users / Groups / Project roles	Explanation
Create Issues	<ul style="list-style-type: none"> Service Desk Customer - Portal Access Project Role (Service Desk Team) Project Role (Administrators) 	Permission to create issues in the project. (Note that the Create Attachments permission is required in order to create attachments.) Includes the ability to create sub-tasks (if sub-tasks are enabled).
Edit Issues	<ul style="list-style-type: none"> Service Desk Customer - Portal Access Project Role (Service Desk Team) Project Role (Administrators) 	Permission to edit issues (excluding the 'Due Date' field — see the Schedule Issues permission). Includes the ability to convert issues to sub-tasks and vice versa (if sub-tasks are enabled). Note that the Delete Issue permission is required in order to delete issues. The Edit Issue permission is usually given to any groups or project roles who have the Create Issue permission (perhaps the only exception to this is if you give everyone the ability to create issues — it may not be appropriate to give everyone the ability to edit too). Note that all edits are recorded in the issue change history for audit purposes.
Transition Issues	<ul style="list-style-type: none"> Service Desk Customer - Portal Access Project Role (Service Desk Team) Project Role (Administrators) 	Permission to transition (change) the status of an issue.
Schedule Issues	<ul style="list-style-type: none"> Service Desk Customer - Portal Access Project Role (Service Desk Team) Project Role (Administrators) 	Permission to schedule an issue — that is, to edit the 'Due Date' of an issue. In older versions of JIRA this also controlled the permission to view the 'Due Date' of an issue.
Move Issues	<ul style="list-style-type: none"> Service Desk Customer - Portal Access Project Role (Service Desk Team) Project Role (Administrators) 	Permission to move issues from one project to another, or from one workflow to another workflow within the same project. Note that a user can only move issues to a project for which they have Create Issue permission.
Assign Issues	<ul style="list-style-type: none"> Service Desk Customer - Portal Access Project Role (Service Desk Team) Project Role (Administrators) 	Permission to assign issues to users. Also allows autocompletion of users in the Assign Issue drop-down. (See also Assignable User permission below)

Assignable User	<ul style="list-style-type: none"> • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to be assigned issues. (Note that this does not include the ability to assign issues; see Assign Issue permission).
Resolve Issues	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to resolve and reopen issues. This also includes the ability to set the 'Fix For version' field for issues. Also see the Close Issues permission.
Close Issues	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to close issues. (This permission is useful where, for example, developers resolve issues and testers close them). Also see the Resolve Issues permission.
Modify Reporter	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to modify the 'Reporter' of an issue. This allows a user to create issues 'on behalf of' someone else. This permission should generally only be granted to administrators.
Delete Issues	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to delete issues. Think carefully about which groups or project roles you assign this permission to; usually it will only be given to administrators. Note that deleting an issue will delete all of its comments and attachments, even if the user does not have the Delete Comments or Delete Attachments permissions. However, the Delete Issues permission does not include the ability to delete individual comments or attachments.
Link Issues	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to link issues together. (Only relevant if Issue Linking is enabled).
Set Issue Security	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to set the security level on an issue to control who can access the issue. Only relevant if issue security has been enabled.

Voters & Watchers Permissions	Users / Groups / Project Roles	Explanation
View Voters and Watchers	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to view the voter list and watcher list of an issue. Also, see the Manage Watcher List permission.
Manage Watcher List	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to manage (i.e. view/add/remove users to/from) the watcher list of an issue.
Comments Permissions		Explanation
Add Comments	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to add comments to issues. Note that this does not include the ability to edit or delete comments.
Edit All Comments	<ul style="list-style-type: none"> • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to edit any comments, regardless of who added them.
Edit Own Comments	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to edit comments that were added by the user.
Delete All Comments	<ul style="list-style-type: none"> • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to delete any comments, regardless of who added them.

Delete Own Comments	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to delete comments that were added by the user.
Attachments Permissions	Users / Groups / Project Roles	Explanation
Create Attachments	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to attach files to an issue. (Only relevant if attachments are enabled). Note that this does not include the ability to delete attachments.
Delete All Attachments	<ul style="list-style-type: none"> • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to delete any attachments, regardless of who added them.
Delete Own Attachments	<ul style="list-style-type: none"> • Service Desk Customer - Portal Access • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to delete attachments that were added by the user.
Time Tracking Permissions	Users / Groups / Project Roles	Explanation
Work On Issues	<ul style="list-style-type: none"> • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to log work against an issue, i.e. create a worklog entry. (Only relevant if Time Tracking is enabled).
Edit Own Worklogs	<ul style="list-style-type: none"> • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to edit worklog entries that were added by the user. (Only relevant if Time Tracking is enabled). Also, see the Work On Issues permission.
Edit All Worklogs	<ul style="list-style-type: none"> • Project Role (Administrators) 	Permission to edit any worklog entries, regardless of who added them. (Only relevant if Time Tracking is enabled). Also, see the Work On Issues permission.
Delete Own Worklogs	<ul style="list-style-type: none"> • Project Role (Service Desk Team) • Project Role (Administrators) 	Permission to delete worklog entries that were added by the user. (Only relevant if Time Tracking is enabled). Also, see the Work On Issues permission.

Delete All Worklogs	<ul style="list-style-type: none"> Project Role (Administrators) 	Permission to delete any worklog entries, regardless of who added them. (Only relevant if Time Tracking is enabled). Also, see the Work On Issues permission.
---------------------	---	---

Using custom permission schemes

If you are a service desk administrator and you want to customize the standard permission scheme, make sure that the roles have the mandatory permissions. See [Customizing JIRA Service Desk permissions](#).

Resolving permission scheme errors

If you encounter any error messages related to your service desk's permission scheme, check out [Resolving JIRA Service Desk permission errors](#).

Installing JIRA Service Desk

If you are migrating from an existing JIRA instance with the JIRA Service Desk add-on, please check out our [Migration Hub](#) first.

Create a basic service desk project for teams that only need a few request types, or create an IT service desk for teams working with change and incident management processes.

Before you get your agents and customers started on a local instance of JIRA Service Desk Server, read the [JIRA A Service Desk release notes](#) for the version that you are installing or upgrading to, then follow these instructions:

1. View the available JIRA Server applications [here](#).
2. Select your JIRA Service Desk Server package.
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 Service Desk instance:

Connecting JIRA Service Desk to other Atlassian products enables a host of new integration features. Learn more below:

- [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.

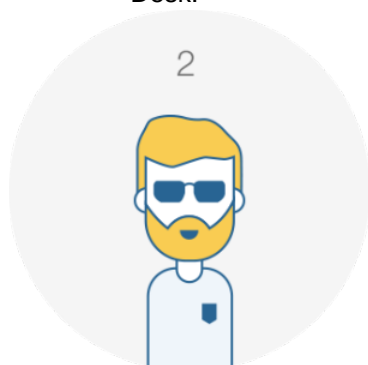
Getting started with JIRA Service Desk

JIRA Service Desk overview

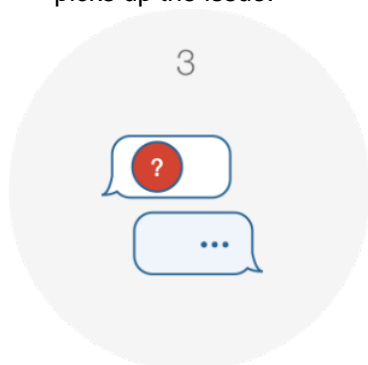
JIRA Service Desk combines the productivity and power of the JIRA platform with an intuitive user experience that allows service teams to focus on the customer. Throughout this tutorial, we will reference the example of a new customer who uses JIRA Service Desk to send requests to his company's IT Team so he can settle into his new role. Here's how the customer and a service desk agent work together to resolve a request using JIRA Service Desk:



1 - Customer needs assistance and submits a request to JIRA Service Desk.



2 - Service desk agent picks up the issue.



3 - Customer and service desk agent discuss the problem.

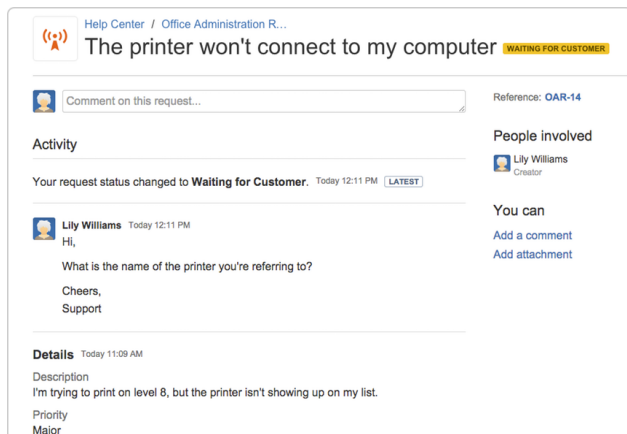


4 - The customer is satisfied and the service desk agent resolves the issue!

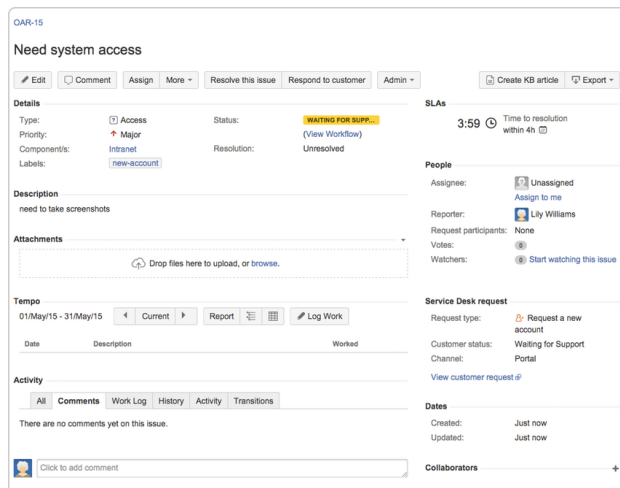
Request vs. issue

Your customers submit a JIRA Service Desk **request** through the customer portal or by email. These requests become issues that your agents work on internally in the JIRA Service Desk agent view.

How customers see a **request** in the customer portal:



How agents see an **issue** in the JIRA Service Desk agent view:



JIRA Service Desk roles

There are three main roles in JIRA Service Desk: administrator, agent, and customer. This guide focuses on the two licensed roles: administrators and agents. The administrator sets up and configures JIRA Service Desk projects. The agent works with the preconfigured service desk projects. Service desk customers are free and do not require a license. You can invite an unlimited number of customers to your service desk projects.

Admin

User with administrative rights for your service desk who can:

- Access all features in JIRA Service Desk
- Add and remove users to and from service desk projects
- Configure the customer portal, request types, queues, reports and SLA metrics
- Perform all tasks outlined in Admin and Agent tutorials

Agent

User who works on and resolves customer requests who can:

- Access the internal service desk interface
- View the customer portal, queues, reports and SLA metrics of assigned service desk projects
- Add, edit and delete customer-facing and private comments on issues
- Manage knowledge base content

Ready to dive into JIRA Service Desk?

I am an admin.

I am an agent.

Getting started for service desk admins

Welcome to JIRA Service Desk for admins! In this tutorial, we'll introduce you to your workspace and walk you through the process of setting up a service desk project for your team of agents and a corresponding customer-facing site (which we call the customer portal). We'll be focusing on basic JIRA Service Desk features and tasks to help you get up and running quickly. By the end of this tutorial, you will have:

- Set up 1 service desk project
- Added 3 agents
- Prepared your customer portal to receive customer requests

Audience:

- Service desk administrators
- Team managers

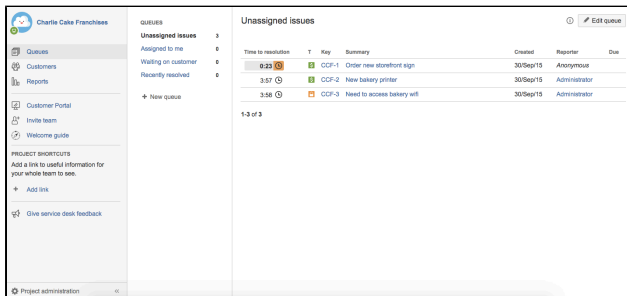
Time: 30 minutes

A quick look at JIRA Service Desk:

Queues



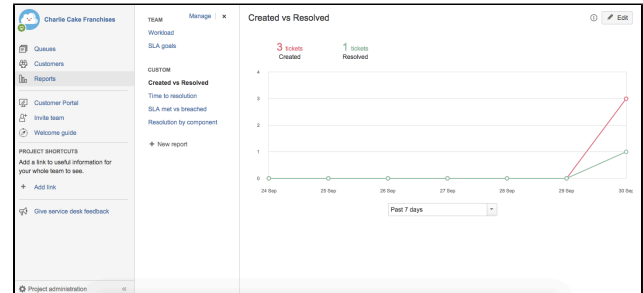
As an admin, you will set up and configure queues for your agents. Your agents will then view and work on issues from the same tab:



Reports



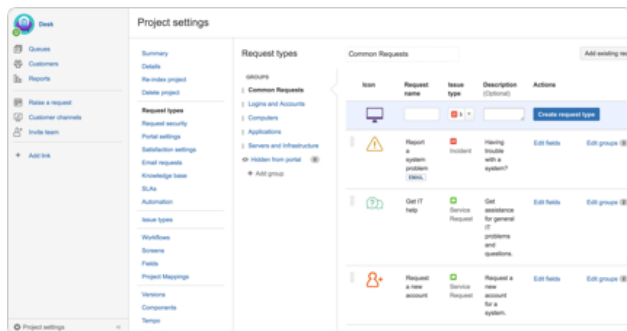
Use the Reports tab to view your team's workload. You can also set up custom reports to track your team's progress in more detail:



Project settings



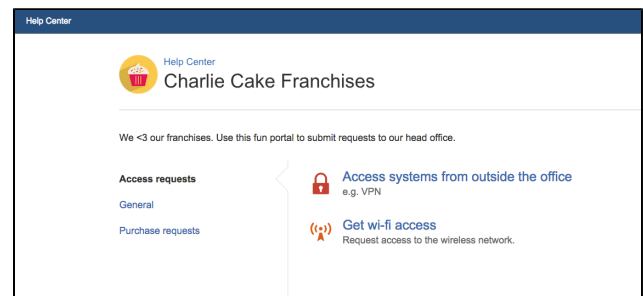
Here, you will set up request types, brand your customer portal, link your service desk to an email account, and manage users:



Customer portal



This link lets you navigate the customer view of your service desk project:



Now that you are familiar with your service desk workspace, you can set up your own JIRA Service Desksite and add your first project.

Let's go!

Setting up your service desk

1. Setting up your service desk
2. Creating service desk request types
3. Making queues for your service desk teams
4. Adding service desk agents
5. Customize

- your service desk channels
- 6. Bring your service desk to the next level
- 7. Introduce customers to your service desk
- 8. Explore a sample project

Let's get your service desk ready to use by setting you up with a JIRA Service Desk Cloud site. Cloud is our hosted offering and will allow you to set up your own site without installing a thing!

If you have an existing Cloud site, [skip ahead](#) to create a service desk project. If your administrator has set you up as a project admin for an existing project, jump to [Step 2](#) to create your request types.

Sign up for a JIRA Service Desk site

Signing up for JIRA Service Desk Cloud will provide you with a fully-functional JIRA Service Desk site for one month.

1. Open [this link](#) in a new tab to view the signup page directly.
2. Follow the signup form steps to enter your site URL and admin username.
3. Once you have completed the signup process, grab a quick coffee (or tea, if that's your preference) — it will take a few minutes for your JIRA Service Desk Cloud site to be created. You will receive an email when your site is ready.

Can't use Cloud?

If you cannot use JIRA Service Desk Cloud, instructions for installing JIRA Service Desk Server are available below.

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

Create a project

JIRA Service Desk comes with default project templates that you can use to suit your team's needs. Create a basic service desk project for teams that only need a few request types, or create an IT service desk for teams working with change and incident management processes. Let's get you set up with a basic service desk project.

1. Open the link just emailed to you to log in to your new site with the administrator credentials you selected.
2. Select **Projects > Create Project** from the the top navigation bar of your site.
3. Select "Basic Service Desk".
4. Name your project. In this example, we'll use the project name "Charlie Cake Franchises". The project key should be automatically populated, but you can change the key if you'd like. If you see options to link another application, leave these options unchecked.
5. Select **Submit** to create your project.

Nice work! You now have a service desk site with one project. You will now learn to set up request

types, which define the requests customers can submit to your team's service desk project.

Next

Creating service desk request types

1. [Setting up your service desk](#)
2. Creating service desk request types
3. Making queues for your service desk teams
4. Adding service desk agents
5. Customize your service desk channels
6. Bring your service desk to the next level
7. Introduce customers to your service desk
8. Explore a sample project

Request types let you define and organize incoming issues so your service desk team can more efficiently help your customers. If you're moving from an existing help desk application, you can add your existing request categories during this step.

If you're setting up service desk request types for the first time:

- Think about how your customer would write a request, for example 'Purchase a new monitor' over 'Submit a hardware request'.
- Break things down into smaller chunks, such as 'Get help with printers' or 'Get wi-fi access'.
- Avoid specialist terminology; think 'I need access' more than 'Deploy SSH key'.

By the end of this step, your project's request type page should look something like this:

Requests vs. Issues

Remember that customers submit *requests* to your service desk and your team picks up the corresponding *issues* to work on internally.

Icon	Request name	Issue type	Description (Optional)	Actions
		Incident		Create request type
	Report a system problem <small>EMAIL</small>	Incident	Having trouble with a system?	Edit fields Edit groups (3) x
	Get IT help	Service Request	Get assistance	Edit fields Edit groups (2) x

Create new request types

Let's go ahead and add two new request types, so you can familiarize yourself with the request type configuration options.

1. In your new service desk project, select **Project settings > Request types**.
2. In the new request type form at the top of the page, change the request type icon and enter the following details for a new "Connect to wi-fi" request type.

The screenshot shows the 'Create request type' form. On the left is a Wi-Fi icon. The 'Request type' field contains 'Connect to wi-fi'. The 'Category' dropdown is set to 'Access'. The 'Field help' text area contains 'Get access to your office's wi-fi network.' A blue 'Create request type' button is on the right.

Select **Create request type** when finished entering your request type details.

3. Create a second request type called "Purchase a monitor" with the following details:

The screenshot shows the 'Create request type' form. On the left is a monitor icon. The 'Request type' field contains 'Purchase a monitor'. The 'Category' dropdown is set to 'Purchase'. The 'Field help' text area contains 'Order a new monitor for your workstation.' A blue 'Create request type' button is on the right.

Select **Create request type** when finished.

Edit the fields your customers see

Now that you have requests in the customer portal, you can prompt your customers to give you the info you need to help them quickly. These simplified fields help customers understand what information they need to provide when submitting a request. Let's add some fields to your request types, so you can collect some additional information.

1. For the "Connect to wi-fi" request type, select **Edit fields**.
2. Under **Visible fields**, click the "Summary" display name and rename it to "What do you need?". Add some placeholder text to the **Field help** to gather useful details from your customers. For example:

Display name	Required	Field help (Optional)	Actions
What do you need?	Yes	e.g. Wi-Fi access for the Houston office.	Update Cancel
Issue field: Summary			

Select **Update** when finished.

3. Select **Add a field** to add the "Priority" field to the request form and select **Apply**.
4. On the **Workflow Statuses** tab, you will see the default JIRA workflow status names displayed on the lefthand side. You can change how these statuses appear to customers by editing the "Status name to show customer" fields as shown:

Workflow status in JIRA	Status name to show customer
Resolved	Completed
Waiting for Support	Waiting for Support
Waiting for Customer	Waiting for Customer

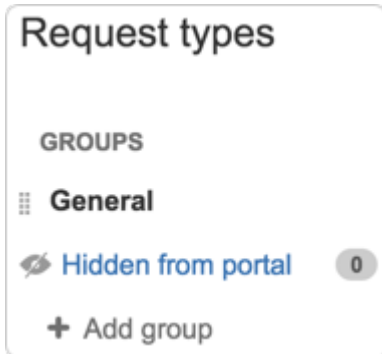
Buttons: Save, Discard unsaved changes

5. Select **View this request form** to see how your changes appear in the customer portal.

Organize your requests with groups

A group is simply a category you can assign to each request type. In the customer portal, your request types are organized vertical tabs based on your groups.

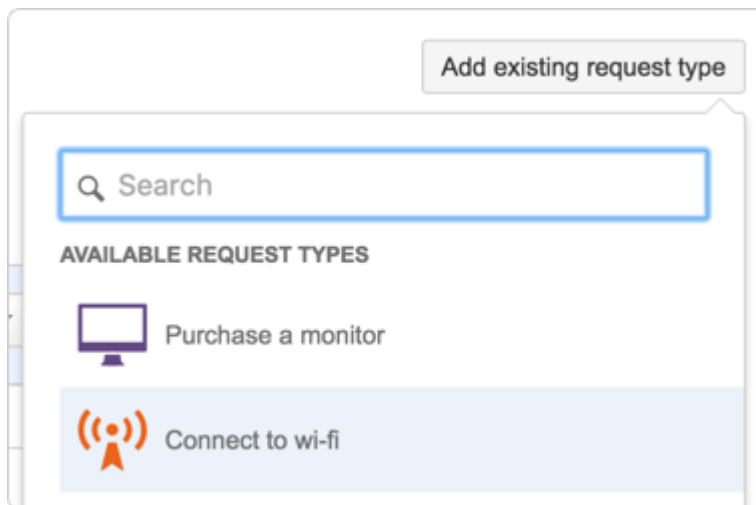
Go to **Project settings > Request types**. You should see your groups in the sidebar:



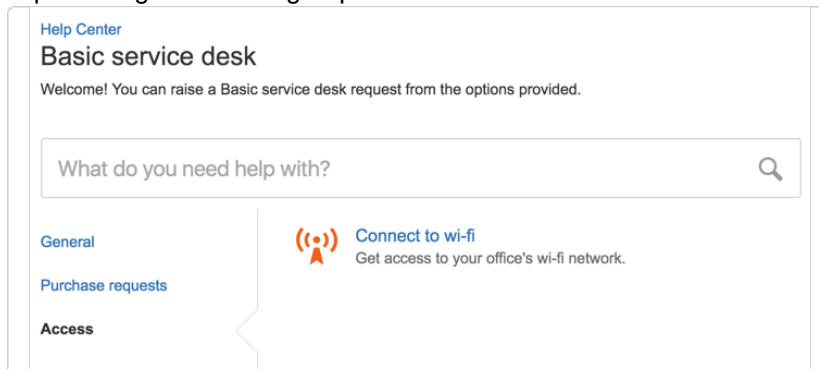
We think groups are helpful if you have *several or more request types*.

Let's add a few groups to help your customers find the request they need:

1. To add request groups, select **+ Add group**. Add two groups for your new request types, "Purchase requests" and "Access".
2. When viewing your "Access" request group, select **Add existing request type** and choose your "Connect to wi-fi" request type:



3. Switch to your "Purchase requests" group in the sidebar.
4. Select **Add existing request type** and choose your "Purchase a monitor" request type.
5. Open the customer portal link from your project sidebar to see your requests organized into groups:



To rearrange the order of how your groups appear in the customer portal, go

back to your project settings and drag and drop the groups in the request types sidebar.

Create a request from the customer portal

1. Keep the customer portal preview open, so you can create test requests from a customer's perspective.
2. Select the "Connect to wi-fi" request type.
3. Enter "Test wi-fi request" in the open field and select Medium priority.
4. Click **Create** to complete your request and view the open request in the customer portal.
5. Click **Close** to exit the customer view and return to your service desk project.

Excellent work! You now have four request types and a new issue in your project. Next, you will learn how to sort these issues into queues, which will allow you to manage your team's workload.

[Next](#)

Making queues for your service desk teams

1. [Setting up your service desk](#)
2. [Creating service desk request types](#)
3. Making queues for your service desk teams
4. Adding service desk agents
5. Customize your service desk channels
6. Bring your service desk to the next level
7. Introduce customers to your service desk
8. Explore a sample project

Your teams will spend the majority of their time working out of the queues you set up. Agents do not have the permissions to add new queues or configure existing ones; however, JIRA Service Desk queues allow you to automatically triage and prioritize issues for them. If you want your team to focus on requests that must be completed by next week, for example, you can set up a queue that only contains requests with a set due date in that week.

Your site comes with preconfigured queues (e.g. "Unassigned issues"), but let's go ahead and create three new queues for your team:

1. From your service desk project sidebar, select Queues.
2. Select New queue and name your first new queue "Access requests".
3. Define the issues you want to appear in this queue by selecting the following drop-down menus: **Type** (select "Access"); **Status** (select "Waiting for Support"), and **Resolutions** (select "Unresolved"):

4. Select the following columns names that will display in this queue from the **More** menu: "Key", "Summary", "Created", "Updated", "Due Date". You can reorder the columns by dragging the name (e.g. "Key") across the column field.
5. Select **Create** to add this queue to your team's workspace.
6. Create two new queues with the following two search queries:

"Completed purchases" for purchase requests that have been successfully resolved.

"Due this week" for requests that must be completed in the next week

7. Reorder your saved queues by clicking and dragging them to their new location.

You now have three new queues in your project! You will next learn how to add agents to your site so you can get your teams up and running with JIRA Service Desk.

Next

Adding service desk agents

1. [Setting up your service desk](#)
2. [Creating service desk request types](#)
3. [Making queues for your service desk teams](#)
4. [Adding service desk agents](#)
5. [Customize your service desk channels](#)
6. [Bring your service desk to the next level](#)
7. [Introduce customers to your service desk](#)
8. [Explore a sample project](#)

There are two default project roles you can assign users to in JIRA Service Desk:

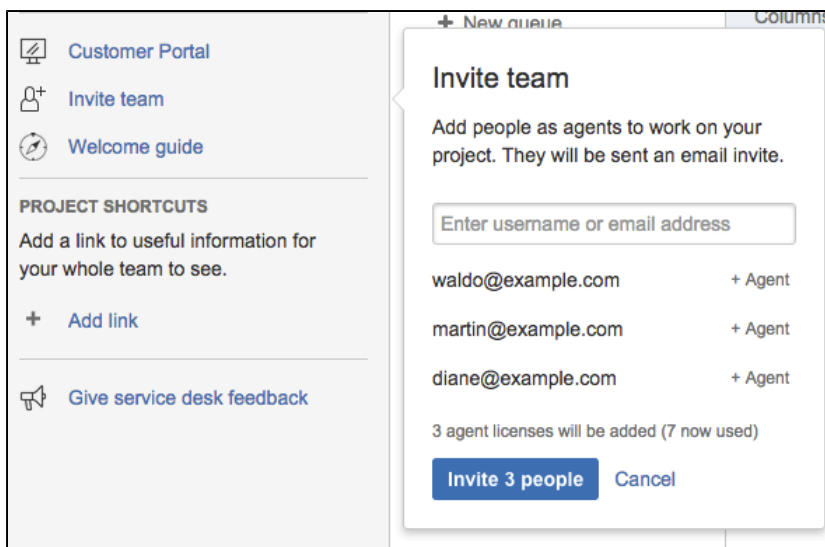
- Service Desk Customers who create requests via email or the customer portal
- Service Desk Team who view and respond to these requests

As the user who created this project, you have been automatically added to the Administrators project role.

Add your agents

Project administrators can only add agents with existing user accounts to their project. If you are a project administrator, you will need to contact your site administrator to add user accounts for new agents. Make sure you're signed in as an administrator for this step, you can invite three new agents to your project - **Diane**, **Martin**, and **Waldo**:

1. In your project sidebar, select **Invite team**.
2. Enter the email addresses for your new agents and select **Invite 3 people**. Your updated agent license count is displayed:



3. These agents will be automatically added to the Service Desk Team role and assigned a JIRA Service Desk license.

Assign issues to agents

Your agents will generally work out of specific queues that have issues automatically triaged into them. Let's test out manually assigning issues in case you ever come across a customer request that you want a certain agent or team to handle.

1. From the **Queues** tab, open one of your test requests by clicking the issue summary or issue key.
2. Select **Assign**.
3. Enter Waldo in the Assignee field and select **Assign**. When Waldo signs into JIRA Service Desk, this issue will appear in his personal queue.
4. Assign another test issue to Diane.

Add your customers

You do not need to add customers to your service desk site during this tutorial but let's check out where you would add them so you're familiar with the needed steps:

1. From your project sidebar, select **Customers**.
2. Select **Invite customers** in the top right corner and enter one or more email addresses.
3. When you select **Send invites**, invited customers will receive an email invitation with a link to your customer portal, where they can complete the signup process.

Public customer signup

You can have your customers sign up for their own accounts (without an individual email invite) by enabling [public signup](#).

You're almost done! You have now added 3 agents to your service desk project and reviewed the process of assigning issues to these agents. You can now customize your customer portal and share it with the rest of your team.

Next

Customize your service desk channels

1. [Setting up your service desk](#)
2. [Creating](#)

- service desk request types
- 3. Making queues for your service desk teams
- 4. Adding service desk agents
- 5. Customize your service desk channels
- 6. Bring your service desk to the next level
- 7. Introduce customers to your service desk
- 8. Explore a sample project

Service desk customers can contact your team in two ways. They can log in and create a request via the customer portal or email a request to an email account that you have linked to your service desk project. Let's finish setting up the customer portal and add an email account so your customers can easily contact your team.

Customize the theme and branding of your customer portal

You can rename your customer portal and add a logo so customers can easily associate this service desk with your team and organization when they create requests.

1. In your service desk project, select **Project settings > Portal settings**.
2. Edit your customer portal name and introduction text by typing in the outlined fields:

Save any edits by selecting



3. Add a customer portal logo by selecting **Use a custom logo for this Customer Portal**.
4. Save the sample image below and select **Choose logo** to upload it:



5. Select **Save logo**.

Link an email account

In addition to creating requests through the customer portal, customers can create requests and communicate with your team by email. JIRA Service Desk Cloud projects come with a default email address, which you can use without having to manage an external email inbox. In this step, you'll link your service desk project to an existing email account used by your team.

1. In your service desk project, select **Project settings > Email requests**.
2. Email requests will be turned off by default, so turn them on now.
3. Select **Add email account** and fill in the requested details. If you use 2-step-verification for Gmail, be sure to generate an [application-specific password](#) when adding your email account details.
4. Once you have linked an email account, look out for the test email that will be sent to your email inbox and the corresponding request that will be created in your service desk project.

Tip:

If you use POP, make sure the email account you choose for this channel has an *empty inbox* so you do not lose any existing emails.

Publicize your service desk

Now that your service desk project is ready to receive requests, you can share the service desk email address (e.g. helpdesk@example.com) and a direct link to the customer portal with your customers.

You can give one or both of the following URLs to your customers.

- The URL to a specific service desk project customer portal. Give this URL to your customers if you've enabled public signup and want them to signup for accounts on their own. The signup link only appears on each individual portal.
- The URL to the global portal where your customers will see all the service desks they have access to. The URL is:

```
http://<computer_name_or_IP_address>:<HTTP_port_number>/jira/servicedesk/customer/portals
```

You can choose to:

- Post a link on your intranet
- Add a hyperlinked button to your web portal
- Email your customers and let them know about the new, easy way to get help!

You've now finished setting up your service desk project! Continue on to learn more advanced tips that will help you better track your team's progress and serve your customers.

Next

Bring your service desk to the next level

1. [Setting up your service desk](#)
2. [Creating service desk request types](#)
3. [Making queues for your service desk teams](#)
4. [Adding](#)

- service desk
agents
- 5. Customize
your service
desk
channels
- 6. Bring your
service desk
to the next
level
- 7. Introduce
customers
to your
service desk
- 8. Explore a
sample
project

Now that you have your basic service desk up and running, you can learn about the following advanced features:

- [Serve your customers and your team better with SLAs](#)
- [Track your team's success with reports](#)
- [Increase self-service with knowledge base integration](#)

Serve your customers and your team better with SLAs

Service-level agreements (SLAs) help you communicate service agreements to your customers and keep track of your team's performance. An SLA consists of a time metric and a corresponding goal or target. As the administrator, you can configure each SLA metric and goal using the JIRA Service Desk SLA designer. SLA information will appear in both the customer-facing request and the internal issue. Your agents can also view SLA goals by going to **Reports > Workload** when they log in to your service desk project. Let's have a quick look at where you can create a new SLA metric.

1. In your service desk project, select **Project settings > SLAs**.
2. Select **New Metric** to create a new SLA metric for your service desk project.
3. For more information, check out [Setting up service level agreements \(SLAs\)](#).

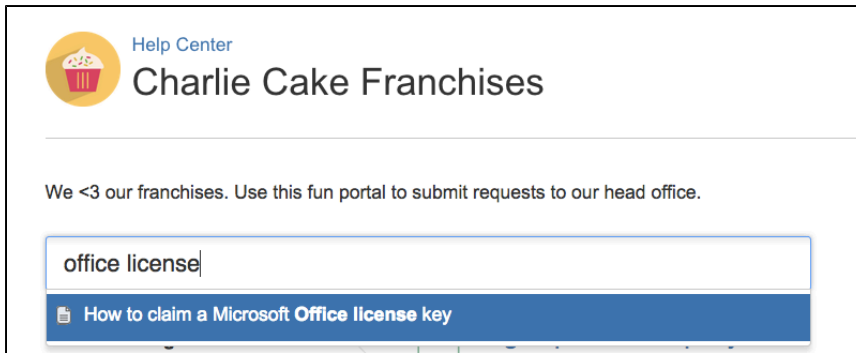
Track your team's success with reports

JIRA Service Desk lets you display selected SLA metrics and goals in interactive reports. Reports can be used to help you visualize your team's performance so you can identify bottlenecks and optimize your team's workload. Your team of agents can then view the read-only versions of your reports to see how they are tracking towards their goals. Let's now have a quick look at the **Reports** tab.

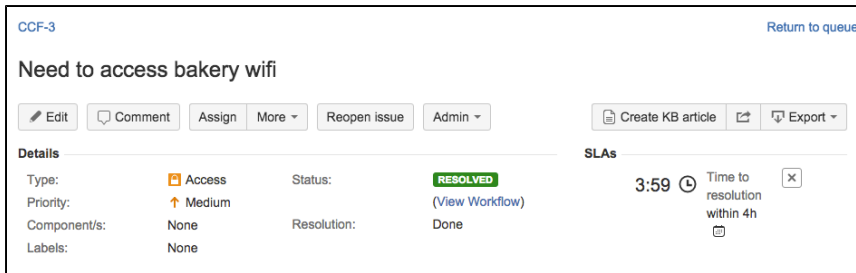
1. From your service desk project sidebar, select **Reports** to view the pre-configured reports in your project.
2. Select **New Report** to create a new report, or simply edit one of the pre-configured reports.
3. For more information, check out [Setting up service desk reports](#).

Increase self-service with knowledge base integration

By connecting Confluence to your service desk project, you can help customers help themselves. Your customers can search for solutions in the self-service customer portal before they even create a request:



Your agents can also take advantage of knowledge base integration by selecting **Create KB article** directly from an issue and saving their customer responses as articles for future reference:



KB articles will be a good resource for new agents in your service desk project and will help prevent existing agents from having to create the same response over again for related issues types.

1. In your service desk project, select **Project settings > Confluence KB**.
2. Choose "Link to a knowledge base" to select the Confluence application and space to link your service desk project to.
3. Check out [Serving customers with a knowledge base](#) to learn more.

You're almost done! We'll now review the ways customers can contact your team and be informed of updates to their requests.

Next

Introduce customers to your service desk

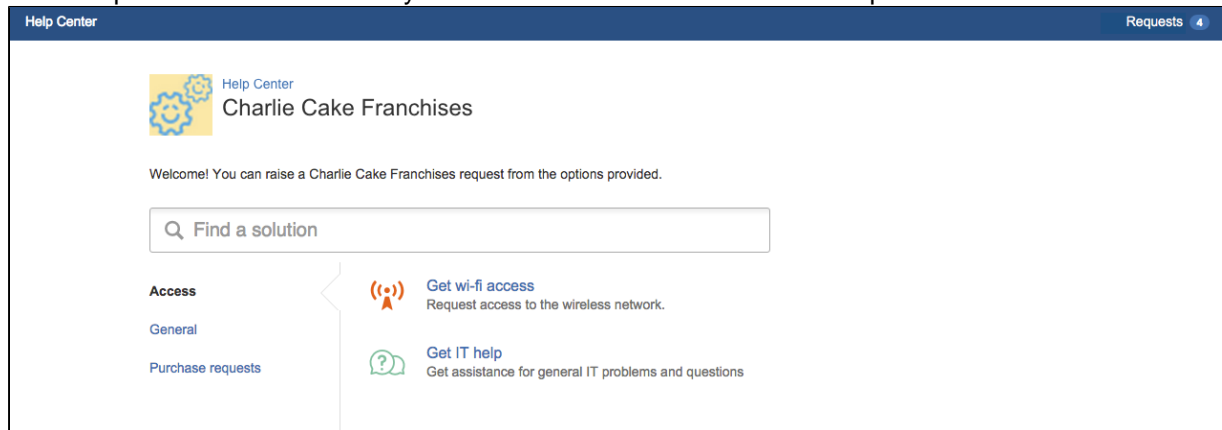
1. [Setting up your service desk](#)
2. [Creating service desk request types](#)
3. [Making queues for your service desk teams](#)
4. [Adding service desk agents](#)
5. [Customize your service desk channels](#)
6. [Bring your service desk to the next level](#)

7. Introduce customers to your service desk
8. Explore a sample project

Now that you have set up your project in a way that serves both your agents and your customers, it's time to show your customers how to start using JIRA Service Desk.

Create requests through the customer portal

1. Visit the customer portal.
2. Pick an option that matches what you need and fill in the details of the request:



Create requests by email

Another way of creating requests is by sending emails to a linked service desk. Ask your service team if they are set up to receive email in their service desk project. If they do, simply email them a request directly and keep the conversation going directly from your inbox.

Create requests in multiple service desks

To send the same request to multiple teams, you have the following options:

- If all of the teams you want to contact have linked their service desk project to an email account, you can easily create the request by sending one email message to all linked service desk email accounts.
- If the teams you want to contact have not all linked their service desk project to an email account, you will need to create the request in each service desk one by one, either through their customer portal or sending emails.

Track and comment on requests

Use the customer portal to see all requests you have created, read comments from agents as they are updated, and check the status of a request. You can add comments and attachments to requests on the customer portal as well.

Another way of tracking requests is through email notifications. You receive email notifications when agents respond to your requests and when the request has been resolved. To add comments to requests, you can simply reply to the email notifications and your reply will be added as a comment to the request.

Congrats! You've completed the Getting started for service desk admins tutorial.

Want to learn more? Check out the home of JIRA Service Desk documentation [here](#).

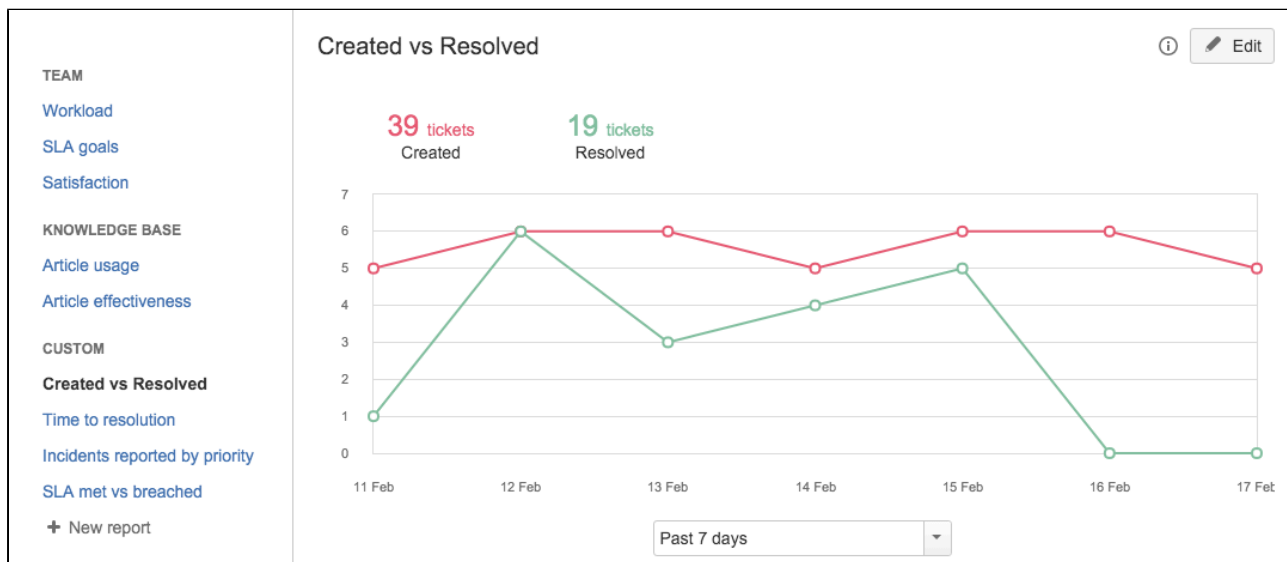
Explore a sample project

JIRA Service Desk comes with sample data to help you explore and learn how to use key features.

When you create a sample project, it gets populated with issues that new team members can use to learn about concepts like queues, SLAs, and generate reports like the one below without fear of affecting any real work.

On this page:

- [Create a sample project](#)
- [Learn about key features](#)

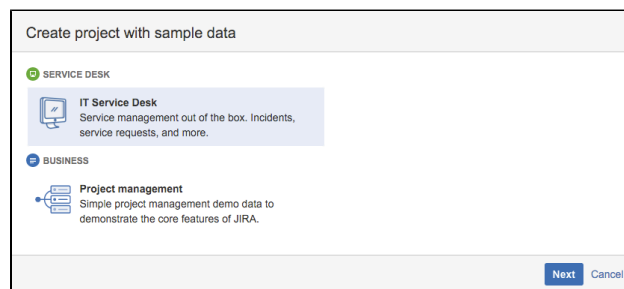


If you are a JIRA Service Desk administrator, we suggest you create and explore a sample project to help you and any new team members explore how a service desk project works

Create a sample project

You need to be a JIRA Service Desk administrator.

1. Go to **Project > Create project**.
2. In the Create project screen, click **Create sample data**
3. In the Create project with sample data screen, select **IT Service Desk** and click **Next**.
4. Enter a name for the sample project.
Tip: If you are creating the project for a specific user, name the project 'Sample - [name user]'. This will make it easier to find and delete later.
5. Click **Submit**.

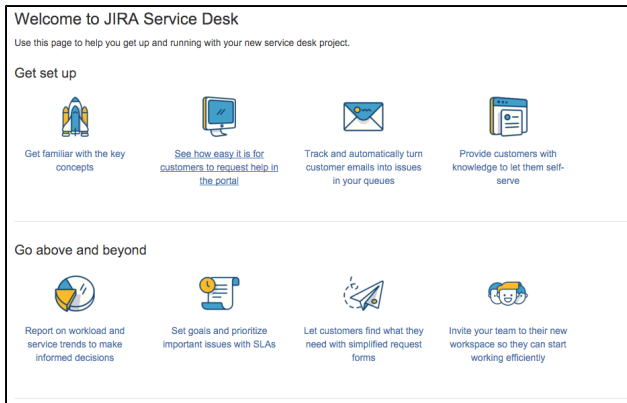


Sample project access Depending on how user access is set up in your JIRA Service Desk version, you may need to [give new users access to the project](#) as well.

Learn about key features

Here are a list of tasks that we highly recommend you have a go at:

- Explore the customer portal - see what your customers will see
- Create a new request and assign it to yourself
- View the queue, edit the issue description, or add a label



- Comment on an issue
- Have a play with available reports.

Finished playing with your sample project?

When a sample project has served its purpose, delete it from the project directory. You need to be a JIRA Service Desk administrator to do this.

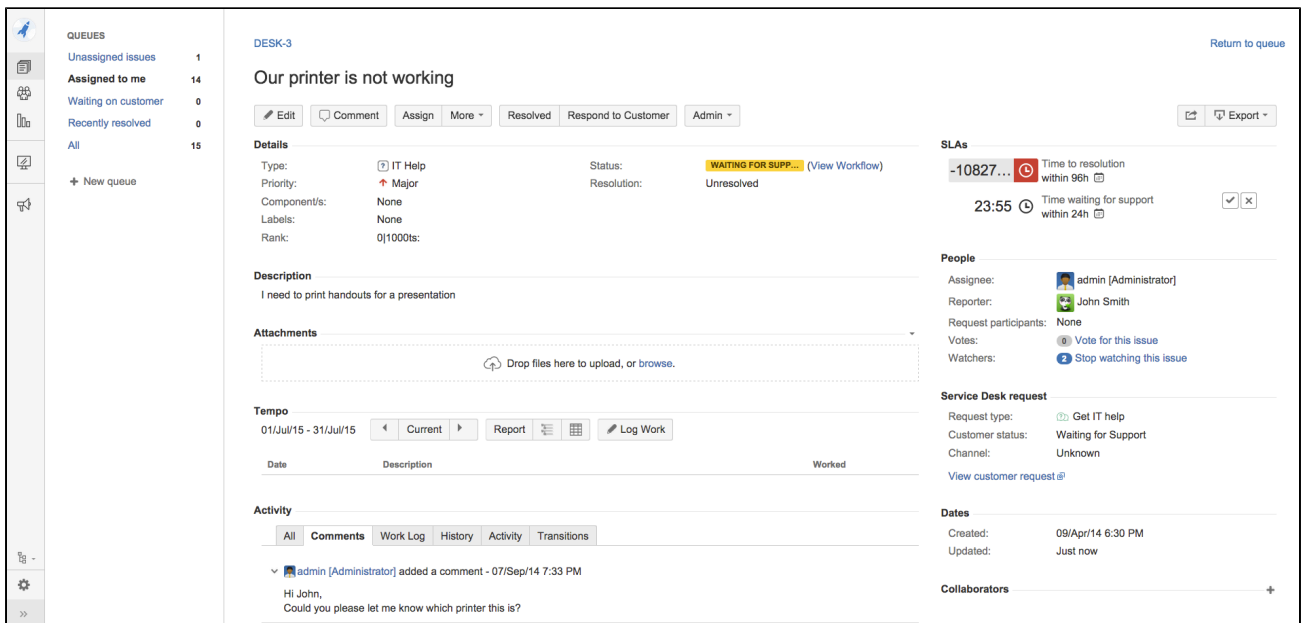
Getting started for service desk agents

On this page, we will introduce you to your workspace and walk you through the process of responding to your customers' requests.

[[Navigate your workspace](#)] [[Work on customer issues](#)] [[Capture knowledge](#)]

Navigate your workspace

Open JIRA Service Desk in your web browser. Take a few minutes to become familiar with the layout:



Queues



Queues display issues submitted by your customers. These issues appear in queues configured by your administrator.

Customer Portal



The customer portal link lets you see and interact with your service desk from a customer's perspective.

Customers



Reports



On the Customers tab, you can search for existing customers in your service desk project, invite new customers (if public signup is enabled), and see how many issues each customer has submitted.

Reports display your team's workload and the Service Level Agreement (SLA) goals configured by your administrator. You can also view any preconfigured service desk reports on this tab.

Work on customer issues

Your administrator has already set up customized queues to help organize incoming customer requests. Please contact your administrator if you need to change a queue's configuration or add a new queue.

Open an issue

1. Select **Queues** and choose a queue to work from (e.g. Assigned to me).
2. Open a customer request by clicking the issue Summary or Key.
3. In addition to being able to edit and comment on a request, you can view a list of actions from the **More** menu. Hover over each action to display a brief explanation:

The screenshot shows the 'New bakery printer' issue page. At the top, there's a header with 'CCF-2' and the title 'New bakery printer'. Below the title are buttons for 'Edit', 'Comment', 'Assign', and 'More'. To the right of these are 'Resolve this issue' and 'Respond to customer' buttons. A 'Details' sidebar on the left lists actions like 'Log Work', 'Attach files', 'Attach Screenshot', 'Add Gliffy Diagram', 'Add Vote', 'Voters', 'Watch Issue', 'Watchers', 'Move', 'Link', 'Clone', 'Labels', and 'Delete'. The main area shows the issue status as 'WAITING FOR SUPP...' with a '(View Workflow)' link. Below this is a 'Drop files to attach, or browse.' section. At the bottom, there's an 'Activity' section with tabs for 'All', 'Comments', 'Work Log', 'History', and 'Activity'. A message at the bottom states 'There are no comments yet on this issue.'

Comment

1. Review the issue and perform the needed task (e.g. grant the customer wi-fi access). Then select **Respond to Customer**. The message you type here displays in the notification sent to the customer.
2. Use the **Internal comment** tab to write your own note or to include another colleague on the issue by using the "@" mention" feature (type @username) and writing your comment.

Attach a file or image

1. Copy and paste, or drag and drop, your attachment anywhere on the issue screen. You'll see the **Add attachments** dialog pop up.
2. Add a comment and select **Share with customer** to send the attachment and comment to the customer, or **Add to issue only** to restrict the attachment and comment to internal users (e.g. other agents).

Resolve an issue

1. Once the customer's request is completed, select **Resolve this issue**.
2. Select a resolution (e.g. Done) and add any further details for the customer or your internal team. All participants on the issue will be notified of its resolved status.

Capture knowledge

If your administrator has linked your service desk with a Confluence space, you can capture your response as a knowledge base article. You can then easily reference this article when responding to a similar issue in the future. KB articles will also appear in the customer portal, directing customers to relevant information before they even finish submitting their requests.

1. Click **create an article** to enter the primary problem/desired outcome (or page title) and select the page template (How-To).

2. Fill out the How-To template and save the page in Confluence. You will see that your issue is linked to this article for future reference.

Nice work! Want to learn more? Proceed to [Working on service desk projects](#) to learn more about what a service desk agent can do.

Administering service desk projects

Welcome to the source of truth for JIRA Service Desk administrator knowledge and power.

- If you're new to JIRA Service Desk, check out [this tutorial](#)
- If you're familiar with JIRA Service Desk, use the search bar below to find any needed information

Search the topics in 'Administering service desk projects':

Automating your service desk

Create automation rules to perform actions in your service desk based on specific triggers and conditions. For example, you can set an automation rule that alerts an agent when a high-priority issue is created. Or, service desk can reopen an issue if your customer comments on it after its been resolved.

On this page:

- [Set up a preset automation rule](#)
- [Edit preset automation rules](#)
- [Create a custom automation rule](#)
- [Disable an automation rule](#)

Set up a preset automation rule

To set up an automation rule:

1. In your service desk project settings, click **Automation** and select **Add rule**.
2. Select a preset rule from the list (see the table below for the available options) and then select **Next**. The rule configuration screen appears.
3. Edit the rule name and description as needed. The rule name appears on the main automation settings page, so changing the name helps you more easily reference what each rule does.
4. Edit and update any fields that appear in red text in the rule's WHEN, IF, and THEN conditions.
5. Select **Save** and you're done.

Your service desk project comes with preset rules that you can use to set up automation. Here are the preset rules that come out-of-the-box to help your team and your customers:

Rule template	What automation does
Transition on comment	Updates the status of an issue after someone comments to: <ul style="list-style-type: none"> • Waiting on Support when a customer comments • Waiting on Customer when your team comments
Reopen on customer comment	Reopens a closed issue when a customer comments
Be aware of urgent issues	Alerts a member of your team via an @mention when a customer submits an urgent request
Keep on top of SLAs	Alerts your team lead via an @mention when a serious issue is about to breach one of your SLAs
Set customer expectations	Lets your customers know when to expect a response from your team based on the priority of their ticket by adding a pre-populated comment to the issue
Update when a linked issue changes	Comment internally on an issue about a change of status for a linked issue
Triage requests sent by email	Update issues received by email with the correct request type, based on keywords present in the request summary or description

Edit preset automation rules

If you want to edit a preset rule, select **Edit** next to the rule in your automation list to see how it's configured.

To edit a rule, select the WHEN, IF, or THEN fields to change your rule's trigger, conditions or resulting actions. Use **Tips for customizing this rule** for suggestions on what to enter in these fields.

If you make changes to the rule, be sure to click **Save** to confirm your edits.

Create a custom automation rule

To create a custom automation rule:

1. In your service desk project settings, click **Automation** and select **Add rule**.
2. Select **Custom rule** from the list and then select **Next**. The rule configuration screen appears.
3. Configure your rule by selecting and defining WHEN, IF, and THEN fields. See the table below for the available options.
4. Select **Save** and you're all set.

Here are the rules sets that are allowed in the automation engine:

WHEN	IF (optional)	THEN
------	---------------	------

Comment is added to an issue	<ul style="list-style-type: none"> • Issue matches a certain filter • Comment visibility is internal or external • User type is a customer or agent • Comment contains a key phrase 	<ul style="list-style-type: none"> • Transition issue to change its position in the workflow • Add comment, either internal or external • Alert user to prompt a specific user or users via an @mention • Edit request type to change the request type (Because request types are mapped to specific issue types, automation isn't able to change issue types. Be sure your request types are the same issue type before applying this rule) • Webhook to send a POST request (see our tutorial)
Issue is created	<ul style="list-style-type: none"> • Issue matches a certain filter • User type is a customer or agent 	
Issue resolution is changed	<ul style="list-style-type: none"> • Issue matches a certain filter • User type is a customer or agent • Resolution change is either set or cleared 	
Status changed	<ul style="list-style-type: none"> • Issue matches a certain filter • User type is a customer or agent • Status change visible to customer 	
A linked issue is transitioned	<ul style="list-style-type: none"> • Link type matches a certain type of link (for example, is related to or blocks) • Issue matches a certain filter • Linked issue matches a certain filter • User type is a customer or agent 	
Request participant added	<ul style="list-style-type: none"> • Issue matches a certain filter • User type is a customer or agent 	
Approval required	<ul style="list-style-type: none"> • Issue matches a certain filter • User type is a customer or agent 	

SLA time remaining Select the SLA and goal status that triggers the event	<ul style="list-style-type: none"> • Issue matches a certain filter
---	---

Disable an automation rule

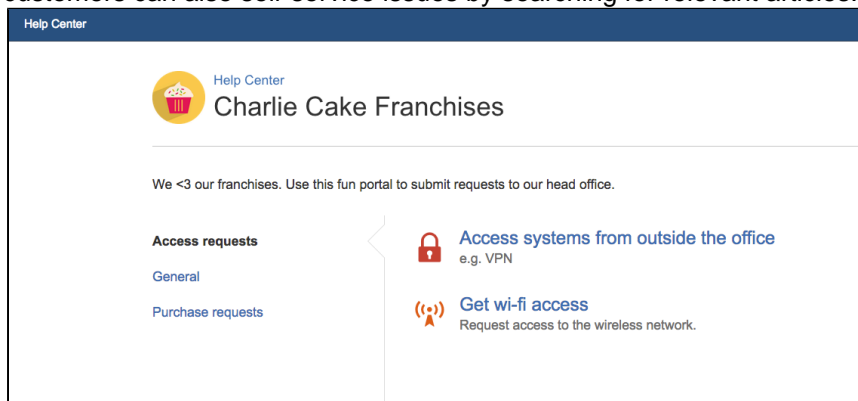
To disable an automation rule:

1. In your service desk project settings, click **Automation**.
2. Select **Edit** next to the rule in your automation list. The rule configuration screen appears.
3. Untick the **Enable rule** checkbox and click **Save**.

Disabled rules appear in your automation list with a **DISABLED** badge.

Configuring the customer portal

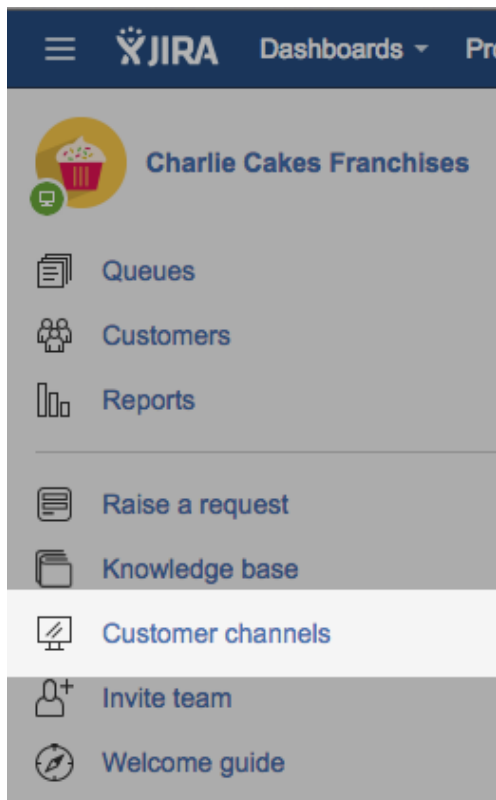
Your service desk project comes with a customizable customer-facing site called the *customer portal* that customers can use to raise and track requests. If you link your project to a Confluence knowledge base space, customers can also self-service issues by searching for relevant articles.



To access your customer portal, click **Customer Channels** in the project sidebar:

On this page:

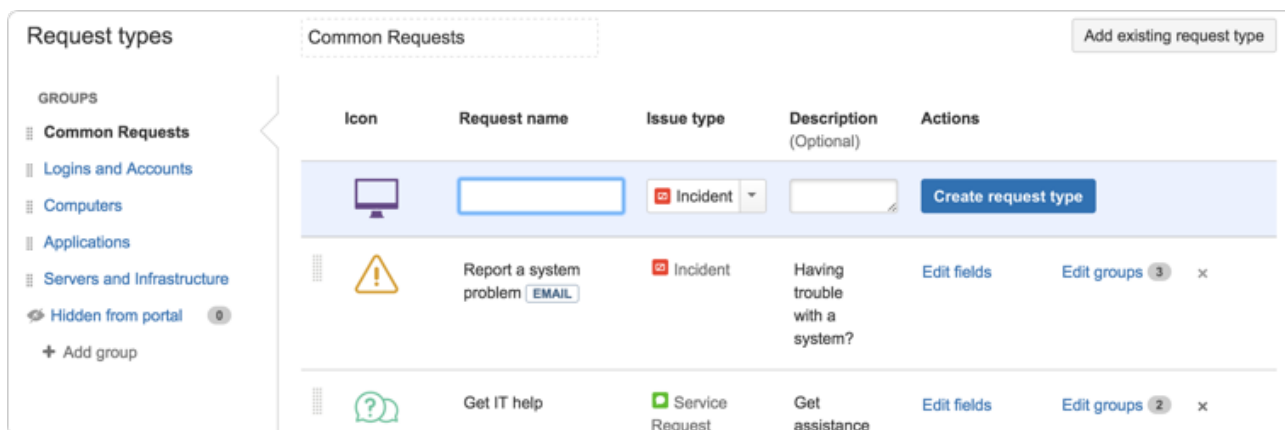
- Set up request types
- Brand your portal
- Add transitions
- Manage access to your portal
- View all portals in your help center



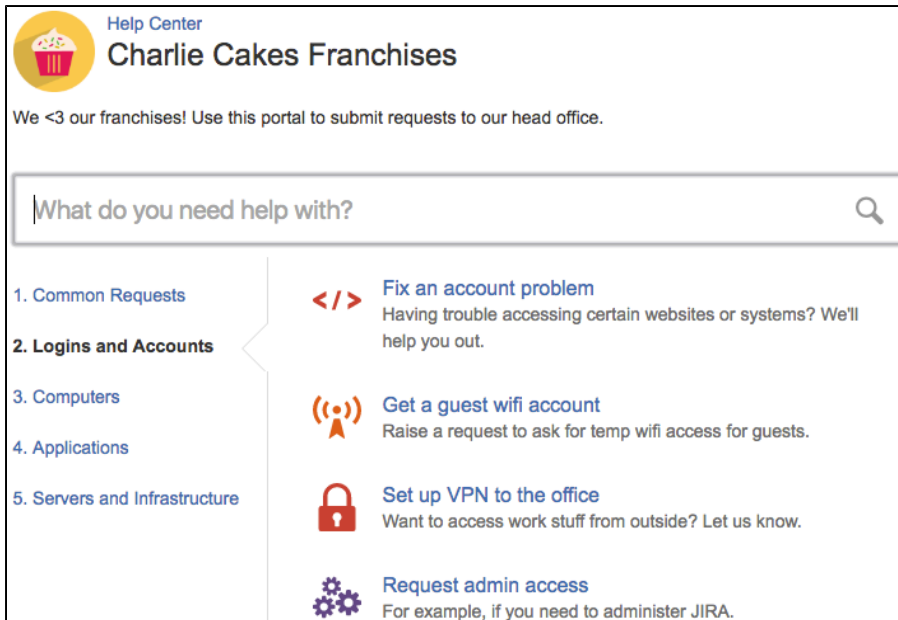
You must be a JIRA Service Desk Administrator or Project Administrator for this service desk project to make the following changes.

Set up request types

You can customize the types of requests that customers raise from the portal. To create and manage request types, visit **Project settings > Request types**.



Service desk includes several request types that address common IT help scenarios. The request types are organized into groups to help customers find what they need on the portal. For example, you can add a "Common Requests" group to help customers get assistance with the most commonly reported issues like system problems or IT support.



Help Center
Charlie Cakes Franchises

We <3 our franchises! Use this portal to submit requests to our head office.

What do you need help with?

- 1. Common Requests
- 2. Logins and Accounts
- 3. Computers
- 4. Applications
- 5. Servers and Infrastructure

- Fix an account problem**
Having trouble accessing certain websites or systems? We'll help you out.
- Get a guest wifi account**
Raise a request to ask for temp wifi access for guests.
- Set up VPN to the office**
Want to access work stuff from outside? Let us know.
- Request admin access**
For example, if you need to administer JIRA.

To learn more about customizing request types, check out [Setting up request types](#).

Brand your portal

You can customize your customer portal to reflect your team and company's brand with the following two steps:

1. In **Project settings > Portal settings**, add a portal logo and a short description to familiarize customers with your service desk:




Portal settings

Customer Portal name Introduction text (Optional)
Charlie Cake Franchises We <3 our franchises. Use this fun portal to submit requests to our head office.

Customer Portal logo

You can now brand your Help Center directly in the Customer Portal.

☐ Do not use a logo for this Customer Portal
☒ Use a custom logo for this Customer Portal

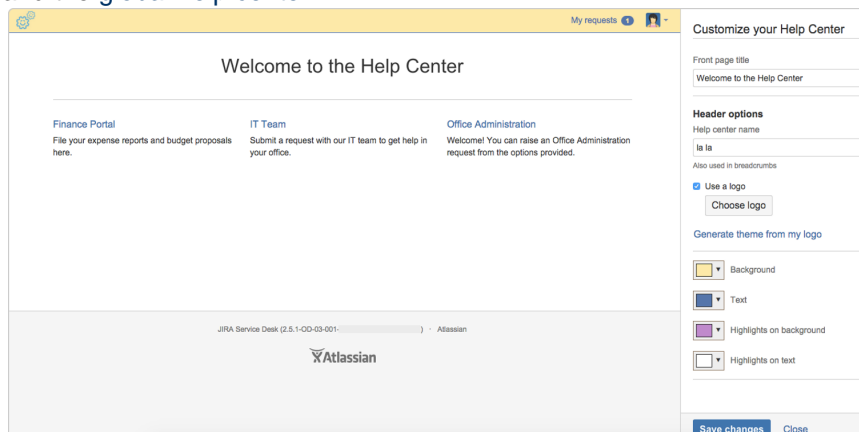
 Choose logo

Save logo Discard unsaved changes

2. Follow the link (or select



> Applications > JIRA Service Desk Configuration) to brand your customer portal header in a live preview mode. The header and branding changes made here apply to all service desk project portals and the [global help center](#):



My requests

Welcome to the Help Center

Finance Portal
File your expense reports and budget proposals here.

IT Team
Submit a request with our IT team to get help in your office.

Office Administration
Welcome! You can raise an Office Administration request from the options provided.

JIRA Service Desk (2.5.1-03-03-001) - Atlassian

Customize your Help Center

Front page title
Welcome to the Help Center

Header options
Help center name
It is
Also used in breadcumbs
☒ Use a logo
Choose logo
Generate theme from my logo


Background
Text
Highlights on background
Highlights on text

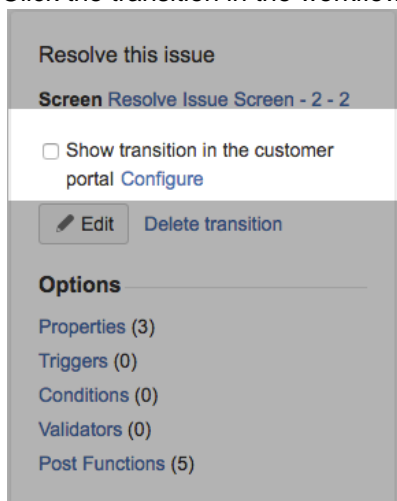
Save changes Close

Add transitions

You can show transitions on the customer portal so that customers can transition requests. For example, say an agent shares a knowledge base article with a customer. If the article solves the customer's problem, then the customer can resolve the request.

To add a transition to the portal, you edit an existing transition in a workflow.

1. In your service desk project, click **Project settings > Workflows**.
2. Click  next to the workflow that contains the transition you want to add to the portal.
3. Click the transition in the workflow, and then select *Show transition on the customer portal*.



Customer transitions behave slightly differently than other workflow transitions:

- Screens don't display on the customer portal. When you add a transition to the portal, you can set a resolution for requests that customers transition.
- When an issue is transitioned from the portal, it bypasses any validators that are defined for the transition.

Note: If it seems like the portal transition isn't working properly, make sure there isn't an automation rule in conflict with the transition.

To learn more about workflows and transitions, see the [advanced workflow](#) configuration page.

Manage access to your portal

You can open your customer portal to allow new customers to create an account and submit a request to your team. If you don't want to have an open portal, you can restrict access to:

- Customers who have an existing account for any other JIRA application or service desk project
- Customers who appear specifically on your service desk project's customer list

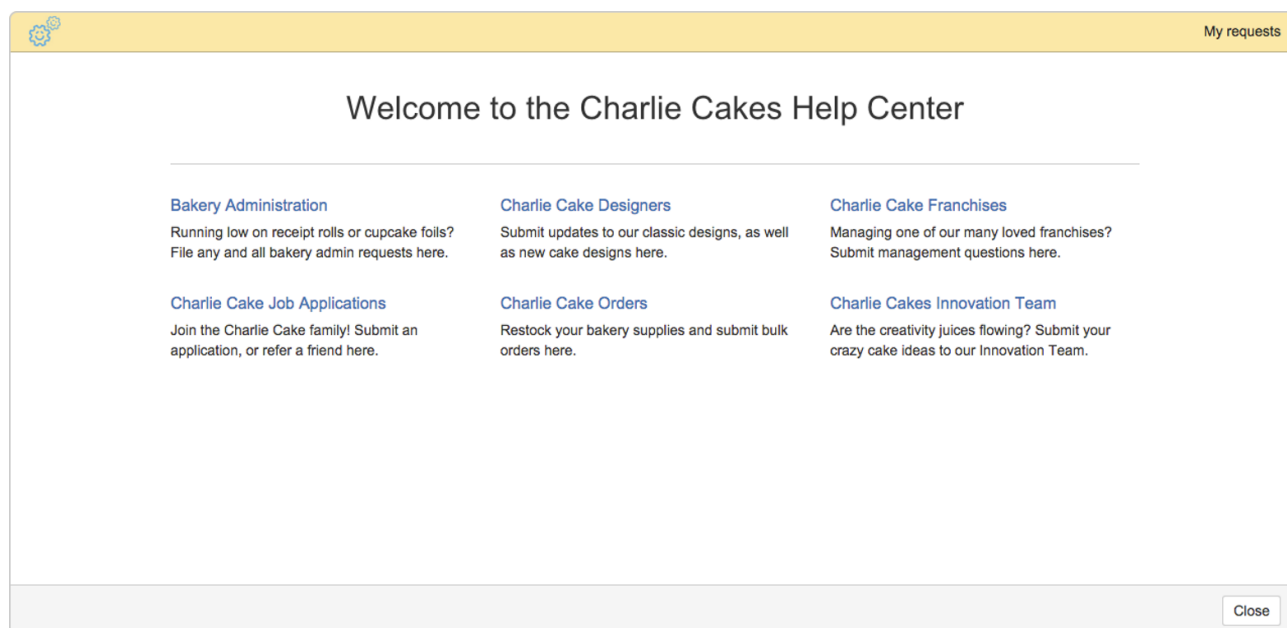
For more information about opening or restricting your portal, see [Managing access to your service desk](#).

The customer portal integrates with [Atlassian Crowd](#), Atlassian's single sign-on (SSO), authentication, authorization, application provisioning, and identity management framework. For information about integrating with third-party SSO providers, see this [page](#).

View all portals in your help center

If your company uses multiple service desk projects (e.g. an IT service desk and an office administration service desk), you can provide your customers with a single URL to find a list of all the customer portals they have access to and the requests created in each one: `http://<computer_name_or_IP_address>:<HTTP_port_number>/jira/servicedesk/customer/portals`

The URL you provide will send customers to what we call the Help Center. The Help Center displays all customer portals generated by service desk projects in a single instance of JIRA Service Desk, as well as the header you [previously branded](#):



To make any changes to the header, or to update the name of your help center, proceed to



> **Applications > JIRA Service Desk Configuration.**

Configuring service desk notifications

When customers submit a request through the customer portal, they receive JIRA Service Desk notifications to keep them informed of the request's progress. Your team of agents and any internal users (e.g. JIRA Software developers) are updated via the default JIRA notification scheme.

On this page:

- [Who receives service desk notifications](#)
- [Configure service desk notifications](#)
- [Set the notification email type to HTML or plain text](#)

Who receives service desk notifications

Your customers	Your team
<p>When a customer submits a request, they receive email notifications as the issue reporter when:</p> <ul style="list-style-type: none"> • the request is created • a comment is added to the request • another participant is added to the request • the request is resolved with a set resolution field • the request is reopened with a cleared resolution field 	<p>When agents work on an issue, they receive email notifications as part of the the project's JIRA notification scheme.</p>

Configure service desk notifications

Two settings impact how notifications work for a service desk project:

- The system level **Notifications** setting controls customer notifications for all service desk projects.

This setting is enabled by default and can be disabled by going to



> **Applications > JIRA Service Desk Configuration.**

- The project level **JIRA notification scheme** controls agent notifications for a service desk project.
 - If the **Notifications** setting for JIRA Service Desk is enabled, agents will receive notifications as part of the JIRA notification scheme. Customers will receive service desk notifications for all issues they're involved with.
 - If the **Notifications** setting for JIRA Service Desk is disabled, agents will continue to receive notifications as part of the JIRA notification scheme; however, customers will only receive notifications if they are working on an issue in a public project.

If notifications are enabled, customers can opt-out of them via the customer portal or email. People who opt out are only notified when the request is resolved.

Set the notification email type to HTML or plain text

As a JIRA administrator, you can set the default email type for service desk notifications. If the default type is set to HTML, dual-encoded notifications are sent, allowing your customers to then select the HTML or plain text view in their mail client. If your customers rely on software that requires plain text or use a plain text mail client, you can change your default setting to plain text and apply this change to new and existing customers.

1. Choose



> **System**. Scroll down to the **User Interface** section and choose **Default User Preferences**.

2. Select **Edit default values**.
3. Change the **Default outgoing email format** to html or text and click **Update**.
At this point, the email format you have selected will only be applied to new service desk customers. If you also want to override the email format chosen by existing service desk customers and agents:
4. Under **Operations**, select **Apply**.
5. Select **Update** to finish applying the email preference to all user accounts.

Managing access to your service desk

When you set up your service desk project, you can allow anyone to sign up for an account and raise a request for your team, or you can restrict your service desk project to a specific list of customers.

On this page:

- [Change your request security settings](#)
- [When to open your service desk](#)
- [When to restrict your service desk](#)
- [Managing request participants](#)
- [Managing customer user field access](#)

Change your request security settings

1. In your service desk project, go to **Project settings > Request security**.
2. Choose one of the options under "Who can raise requests". The security options you can select are explained in more detail below.

When to open your service desk

As an example of when to open your service desk project, an IT service desk is usually open to all the employees in an organization, so everyone can access it and create requests. In this open service desk scenario, customers can create an account on the customer portal or email requests to your service desk email channel to have an account created automatically. To open your service desk, your administrator must first enable [public signup](#). You can then select the first option under "Who can raise requests" on the request security page.

When to restrict your service desk

If public signup is not enabled, you have the option to partially restrict your service desk to service desk customers and JIRA application users from any type of project. This option prevents people from signing up themselves and emailing your linked email channel without an existing account.

To partially restrict your service desk, select the second option under "Who can raise requests" on the request security page.

In another example, for a service desk that handles contractors' leave requests, you might want to make it only available to your contractors so that the rest of your staff do not get confused about where to put in leave requests. Service desks like this one are restricted service desks and only customers you add to your project's Customers list can create requests.

To restrict your service desk, select the third option under "Who can raise requests" on the request security page.

Managing request participants

You also manage adding request participants on the **Request security** page. Make the selection between **Agents only** or **Agents and Customers**. Check out the [Adding request participants](#) page for more information on what how to add request participants, and what a request participant can do.

Managing customer user field access

In certain situations you may want to allow your customers to browse for users (this includes licensed JIRA users and your other customers). For example, when using approvals you may want your customer to be able to browse the users on your system by their user name or email, so that it's easier for them to select the appropriate approver. Browsing for users works when the customer begins to type a user name or email in a user field, any matches are displayed in a drop-down list, which the customer can then select from. If you don't want your customers to be able to browse your JIRA instance's users (this includes all licensed JIRA users and other customers), then make sure you select the appropriate setting.

Receiving requests by email

If your customers prefer to open and work on requests from the comfort of their email inboxes, you can enable email requests to receive all customer requests in JIRA Service Desk. Enabling email requests will help your team focus on your customers, instead of having to worry about missing requests or checking multiple inboxes.

Here's how it works:

1. A customer emails a request to your linked service desk account
2. An agent comments on the request in service desk, which sends the customer an email notification.
3. The customer replies to these email notifications until the request is resolved. All customer replies are automatically added as comments on the corresponding issue in service desk.

On this page:

- [Before you start](#)
- [Add an email account](#)
- [Choose a request type](#)
- [Verify your linked email account](#)
- [Prepare customers for email greatness](#)
- [Email usage notes](#)

Before you start

- Make sure you have the Administer Projects permission.
- Enable [public signup](#) or manually [add customers](#) to your service desk project to ensure that you receive new customer requests.
- Set up a suitable [request type](#) with **Summary** and **Description** as required visible fields. Any other fields must be optional.
- Know which emails from your mail client will be [processed](#).

JIRA Service Desk uses a built-in email processor to manage incoming emails in service desk projects. It's purpose-built for service desk projects and works differently from [JIRA mail handlers](#). Issues created via JIRA email handlers don't show up as service desk customer requests. For this reason, we don't recommend using a JIRA mail handler for service desk projects.

Add an email account

Open your service desk project and proceed to **Project settings > Email requests**. Select **Add an email address**. Easy enough. Choose your email service provider and enter the requested details before selecting **Next**.

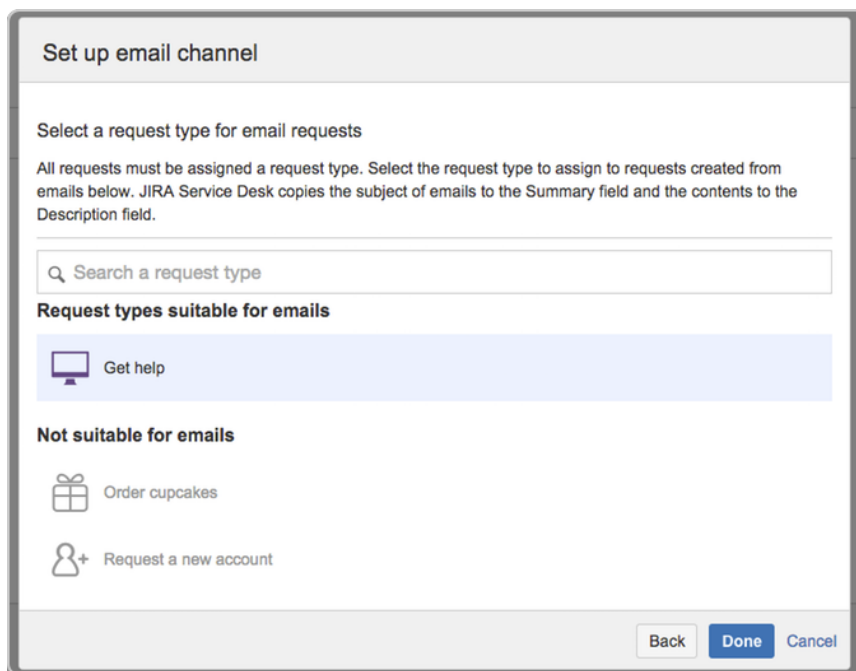
If you have two-step verification enabled for your Gmail or Yahoo! account, you need to set up an [application-specific password](#).

Choose a request type

When a customer emails your service desk, a corresponding request is created with the following two fields:

- Summary (from the email subject line)
- Description (from the email content)

In order to use the email channel, you therefore need to have at least one request type in your project with the Summary and Description fields – we call these types of requests "suitable for emails". Associating email requests with a suitable service desk request type ensures that the emails are successfully filtered into your service desk queues. In this example, we have one suitable request type ("Get help"):



The screenshot shows a modal window titled "Set up email channel". Inside, there's a section "Select a request type for email requests" with a sub-note: "All requests must be assigned a request type. Select the request type to assign to requests created from emails below. JIRA Service Desk copies the subject of emails to the Summary field and the contents to the Description field." Below this is a search bar labeled "Search a request type". Underneath, there are two sections: "Request types suitable for emails" and "Not suitable for emails". In the "suitable" section, "Get help" (with a monitor icon) is selected and highlighted with a blue background. In the "not suitable" section, there are two items: "Order cupcakes" (with a gift icon) and "Request a new account" (with a person icon). At the bottom right of the modal are three buttons: "Back", "Done" (in blue), and "Cancel".

Select a suitable request type for your email channel and select **Done**.

Verify your linked email account

Once you have chosen a suitable request type, JIRA Service Desk will send a test email and create a corresponding test request in your service desk project. Head on over to the **Queues** tab to find the new request:

CC-10

[JIRA] Requests from test@example.com are now set up and ready to use

[Edit](#) [Comment](#) [Assign](#) [More](#) [Resolve this issue](#) [Respond to customer](#) [Admin](#)

Details

Type:	IT Help	Status:	WAITING FOR SUPP... (View Workflow)
Priority:	Major	Resolution:	Unresolved
Component/s:	None		
Labels:	None		

Description

Congratulations! The email channel for your service desk has now been set up successfully.

This request comes from the test email that is automatically sent to your email address test@example.com

When you add customer-visible comments, your customers will receive an email update. When they reply to the email, their reply will be added as a comment to the request.

If this email remains in the inbox for more than 5 minutes, then the incoming emails are not being properly processed.

Activity

[All](#) [Comments](#) [Work Log](#) [History](#) [Activity](#) [Source](#) [Reviews](#)

There are no comments yet on this issue.

[Comment](#)

New messages sent to your linked email account will now appear as service desk requests in your project. For more information about what emails are processed by JIRA Service Desk, expand the option that applies to you:

Emails using POP

▼ JIRA Service Desk looks for messages in your inbox that have...

1. The "Deleted" flag set to false, and
2. Been received after your email account and service desk project have been successfully linked.

To link your email account with a service desk using POP, make sure that your email inbox is empty by moving the existing messages to another folder, archiving them, or deleting them. Starting with an empty inbox ensures that you do not lose emails unintentionally, as POP emails are deleted after they are processed by JIRA Service Desk.

Emails using IMAP

▼ JIRA Service Desk looks for messages in your inbox that have...

1. The "Deleted" and "Seen" flags set to false, and
2. Been received after your email account and service desk project have been successfully linked.

If you use IMAP, emails are marked as read (not deleted) after they are processed by JIRA Service Desk. If you want existing messages to be pulled in by JIRA Service Desk, you can move them back to your inbox and mark them as unread after the connection has been established.

Prepare customers for email greatness

Before sharing your linked email account with your customers, you will want to confirm whether you have an [open or restricted service desk](#). New customer email requests can bounce if you have a restricted service desk, and that's no fun. Expand the statement that applies to you below to make sure your customers are ready to use your new email channel:

▼ I have an open service desk...

Great! New customers can create requests right away by emailing your linked service desk email account. A corresponding customer account will be created based on the new customer's email address. Customer accounts do not count towards your service desk license.

▼ I have a restricted service desk...

Email requests will not be processed if your customers don't have existing service desk accounts. Simply create [new customer accounts](#) (or send customer invitations) before telling new customers to email your service desk.

Email usage notes

- Note that you can only link one email account to your service desk project. If you use more than one email address to interact with your customers, you might be able to set up forwarding rules or aliases to receive requests in the email linked to your service desk project. You will need to configure any forwarding rules or aliases in your email client.
- If you are a JIRA Administrator, you can refer to [Managing the email channel](#) to learn more about global mail settings.
- If you encountered any issues during the email setup process, check out some common errors and resolutions [here](#).

Managing the email channel

After you [set up your email channel](#), you can control when JIRA connects to your mail server and filters relevant emails into your service desk projects. You can also view logging information directly in JIRA to check the status of your mail server connection.

On this page:

- [Manage global mail settings](#)
- [Manage the email channel for multiple service desk projects](#)

Manage global mail settings

There are two global mail settings - email puller and email processor - that are used only by JIRA Service Desk and do not impact any email settings you have set up for JIRA. Email puller connects to your mail servers every minute and pulls the email data into the database. Emails with attachments larger than 25MB will not be pulled. Email processor filters the emails (e.g. to remove auto-replies and spam) using information stored in the database.

You can access these settings by going to



> **System** > **Global mail settings**.

Manage the email channel for multiple service desk projects

JIRA administrators can manage the email channels for all of the service desks in the system. To manage the email channels, Choose



> **Applications**. Scroll down to the **JIRA Service Desk** section and choose **Email requests**.

On the **Email requests** page, you can choose how your service desk formats emailed comments. You can also check your email channel connections and view the logs for each channel. Note that logging information older than 6 months is deleted daily.

Troubleshooting issues with the email channel

This page contains information about the errors you might run into when setting up the email channel for your service desk. See the [Resolving errors](#) section below for more information about generating an application-specific password (e.g. if 2-factor authentication is enabled for your email account), resolving email connection issues, and ensuring that customers who raise requests by email are correctly notified.

Checking the connection

To troubleshoot email channel issues, the first thing to do is to check the connection between JIRA Service Desk and your email account. You will see error messages that show you why the email channel does not work for your service desk.

To check the connection:

1. Choose



> **Applications**. Scroll down to the **JIRA Service Desk** section and choose **Email requests**.

2. Select **Test**.

Resolving errors

The following table describes the common errors and provides information about how to resolve them when available.

Symptom	Description and resolution
<p>Setting up the email channel</p> <p>Error message:</p> <p>The email address you entered is currently used by another project's email channel. Please choose another email address. Check out our troubleshooting docs for help resolving the issue.</p>	<p>You can only configure one email address per service desk project in your JIRA Service Desk Cloud site or Server instance. If you try to use the same email address to set up an email channel in another service desk project, you'll receive this error message.</p> <p>You can also receive this error message if you are trying to use multiple email aliases that point to the same email account for multiple service desk email channels.</p> <p>To resolve this:</p> <ul style="list-style-type: none">• Choose another email address for the email channel you're setting up or for the one that already exists.

<p>Setting up a Gmail account</p> <p>Error message:</p> <p>Unfortunately JIRA couldn't connect to the mail server. Here is what the mail server said: "[ALERT] Please log in via your web browser: http://support.google.com/mail/accounts/bin/answer.py?answer=78754 (Failure)</p>	<p>If you have two-factor authentication enabled for your Gmail account, you'll likely receive this error when entering your login details. Alternatively, JIRA Service Desk checks email accounts every minute, causing Gmail to suspect inappropriate usage of this account and lock it for security reasons.</p> <p>To resolve this:</p> <ul style="list-style-type: none"> • Create an application-specific password for JIRA Service Desk in your Gmail account settings. Details can be found here.
<p>Setting up a Yahoo! account</p> <p>Error message:</p> <p>Unfortunately JIRA couldn't connect to the mail server. Here is what the mail server said: "[AUTHENTICATIONFAILED] (#MBR1240) Please verify your account by going to https://login.yahoo.com"</p>	<p>If you have two-factor authentication enabled for your Yahoo! account, you'll likely receive this error when entering your login details.</p> <p>To resolve this:</p> <ul style="list-style-type: none"> • Create an application-specific password for JIRA Service Desk in your Yahoo! account settings. Details can be found here.
<p>Microsoft Outlook, POP3</p> <p>Error message:</p> <p>Unfortunately JIRA Service Desk couldn't connect to the mail server. Here is what the mail server said: "STAT command failed: Exceeded the login limit for a 15 minute period. Reduce the frequency of requests to the POP3 server."</p>	<p>JIRA Service Desk checks email accounts every minute. Microsoft Outlook might suspect inappropriate usage of this account and lock it for security reasons.</p> <p>To resolve this:</p> <ul style="list-style-type: none"> • Use IMAP.

<p>Gmail accounts, POP3</p> <p>Requests are created from archived messages.</p>	<p>When JIRA Service Desk checks your email accounts for new messages, it looks in the inbox folder.</p> <p>Gmail uses labels to classify messages into categories and has these folders: inbox, Sent Mail and Trash (or Trash). This means that the archived messages are still considered in the inbox folder. With POP3, JIRA Service Desk is not able to identify archived messages by label and therefore still brings them in as requests.</p> <p>To resolve this:</p> <ul style="list-style-type: none">• Use IMAP.
---	--

Customers send emails to create requests, but no requests are created and customers do not receive any notifications.

This problem could be due to one or more of the following causes:

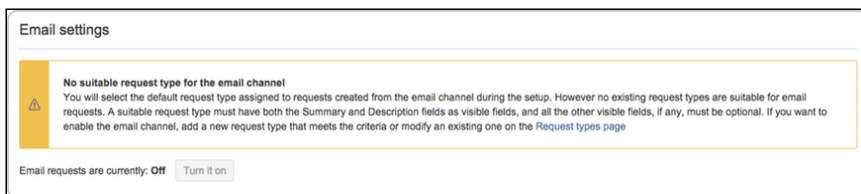
- The connection to the email account has failed.
- You do not have a public signup configured and the customer does not have a user account in the system. Every customer must have an account before they can create requests in a Jira Service Desk.
- The default request type for the email channel is unsuitable for the email channel.

▼ [Learn more](#)

A suitable request type for the email channel must be configured in the **Summary** and the **Description** field as visible fields. Any other fields must be optional ones.

To troubleshoot this issue and resolve it

1. Check the connection as described previously on this page.
2. Check if user accounts exist for your customer. If not, create user accounts for your customers. For instructions, see [Setting up service desk users](#). You can also configure a public signup.



This message appears on the **Email requests** page and prevents you from turning on the email channel.

To resolve this:

- Head to **Project settings > Email requests**
- Add a new request type (or choose an existing one)
- Select **Edit fields**
- Make sure both the Summary and Description fields are added and marked as **Required**.
- Save the request type and head back to **Project settings > Email requests**

Setting up approvals

JIRA Service Desk allows you to add an approval step to any status on your project's workflows, and this step means that when a request hits this status, an approver will be notified that they need to approve or decline the request from the customer portal. If you set up the approval step so that the request can *only* be approved or declined, the approver needs to make their decision to progress the request. If the status has more than two outgoing transitions, only two of these can be used for Approve or Decline, the additional transitions can be actioned by an agent. This is useful in the case of exceptions, for example if a request is urgent and can't wait for the approver, it's useful to have another method to progress the request.

On this page:

- [Setting up an approval step](#)
- [How do I make this even more awesome for my customers?](#)
- [Approval FAQs](#)

Approvers view in the customer portal

Here's how it works:

- A customer creates a request on the customer portal, and selects the approver required by entering the user name or email address
- The approver receives an email notification that they have a pending approval when the request enters the approval status, they can view and action the request through the customer portal
- The approver either declines or approves the request, and can add an optional comment
- If declined, the request moves to the next status, and if a comment is added the customer will receive an email notification
- If approved, the request moves to the next status in the workflow and an agent is able to work on it, and if a comment is added the customer will receive an email notification

Alternatively, you can hide the approver field from the customer, and set up a defined list of approvers that are required for that request type.

If you set up your approval step on a status with only two outgoing transitions, they will be used for Decline and Approve. In this case, agents can view requests that require approval, and can modify the approver if required, but they can't change the status until the approver has actioned the request. If you set up the approval step on a status which has more than two transitions, an agent will be able to transition the request using any of the other transitions that aren't defined as the Approve or Decline transitions. This means the approval step is not enforced.

In certain situations, you may even want to add additional approval steps, for example if a request needs to be approved by your manager first, and then approved by your finance department. Below are some examples of how you might use approvals:

Request type	Approval field	Approver	Notes
Office equipment	Visible on customer portal	Customer selects approver.	The customer should select their manager for approval, as each customer could have a different manager.
Computer software	Hidden from customer portal	Set list of approvers.	You set the list of approvers, this could be several members of your finance team.
Flights	Two approval steps, one visible, one hidden	First approver is selected by customer, second approver is a set list.	The customer can select the first approver, which could be their manager who approves the business case for their trip. The second approver is from a set list (maybe a finance team) that approves the payment of the flights.

Setting up an approval step

To set up an approval step on a workflow for your project, you need to have the JIRA administrator [global permission](#) .

Here are the steps to get approvals working for your project:

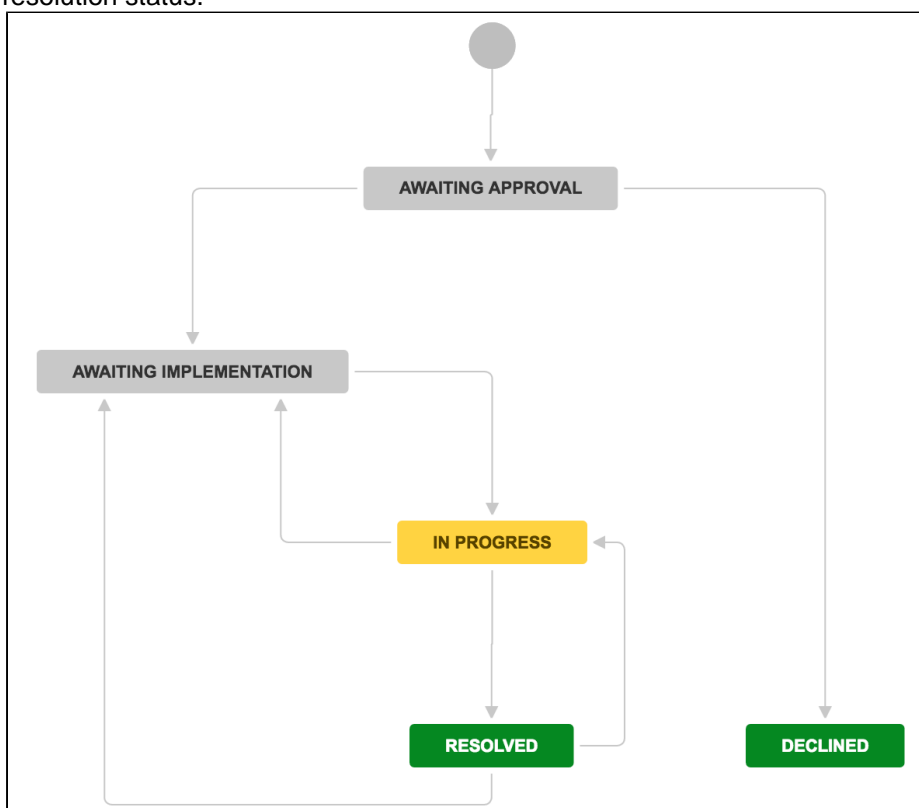
- JIRA Service Desk creates the **Approvers** custom field automatically. If you want to use another field, make sure you have a user picker custom field available in your JIRA instance and on the screens used by your project - this is used on the approval step
- Add the same user picker custom field to your request type if you want your customers to choose the approver - this will provide the field for your request that allows an approver to be selected
- Configure the approvals step on the workflow - this allows you to decide if you require one or multiple approvers, and what happens when the approval is declined or approved

Detailed steps on [setting up an approval](#) are available in the JIRA administration documentation.

How do I make this even more awesome for my customers?

You can make the approval process more awesome by making some simple customizations to your service desk project, and that means both you and your customers have a better experience.

- Make sure you make the name of your multi-user picker custom field customer friendly on your request, and add a useful help tip, like how many approvers may be required. This will help ensure your customers provide all the correct information first time. Read up on [customizing the fields of your requests](#) for more information on how to do this.
- When the approval of a request is declined, consider closing the request straight away. This means that if you're setting up a custom approval step, you should ensure that the transition you select for decline leads to a status in the Done category, and the transition has a post function to set the resolution status.



Workflow showing Declined transition leading to Declined status in the Done category

- It's good practice for your approvers to add a comment telling the customer why their request has been declined, and what their next steps should be, for example the customer should open a new

request and provide more information regarding their requirements. A customer won't be automatically informed when a request has been declined or approved, but they will be notified of any comments added to the request.

Approval FAQs

Why isn't the field I added to my approval step showing on my customer portal?

Make sure you've added the user picker custom field that's used in the approval step on the workflow to your [request type](#), and it's visible to customers. If it's still not showing, or it's not listed as a field you can add to your requests, check with your JIRA administrator to ensure the field is still available on your project's issue screens.

Declined requests are still showing as open in my service desk project, but they're status shows as Done.

A request will only show as closed when the resolution has been set, so the transition you use to decline the request needs to also set the resolution field. You can achieve this by adding a post function to the transition in your workflow that sets the resolution field of your request. You read up more on post functions on the administration [advanced workflow](#) page.

How do I add, edit or remove approvers on a request?

If you need to change the approvers on a request for any reason, you need to edit the user picker custom field that was added in the approval step. View the request in your service desk project, and the editable user picker custom field is displayed in the **People** section of the request. You can make any changes you need to make inline. Note that there is also an **Approval** section that lists all approvers, however you can't make any edits here.

The screenshot displays a JIRA Service Desk request interface. At the top, the title is 'Upgrade the SC3 server to JIRA 7.1.3'. Below the title are buttons for 'Edit', 'Comment', 'Assign', 'More', 'Third', and 'Admin'. The 'Details' section shows 'Type: IT Help', 'Priority: Medium', 'Component/s: None', 'Labels: None', 'Status: NEEDS APPROVAL (View Workflow)', and 'Resolution: Unresolved'. The 'Description' section contains the text: 'We've tested JIRA 7.1.3 and it's passed all our tests. Please proceed with the upgrade.' The 'Attachments' section has a dashed box with the text 'Drop files to attach, or browse.' The 'Activity' section shows 'All', 'Comments', 'Work Log', 'History', and 'Activity' tabs, with a message 'There are no comments yet on this issue.' The 'SLAs' section shows '3:58' and 'Time to resolution within 4h'. The 'People' section lists 'Assignee: Unassigned (Assign to me)', 'Reporter: Elaine Gould', 'Request participants: None', 'Approval: Frank Smith', 'Votes: 0 (Vote for this issue)', and 'Watchers: 0 (Start watching this issue)'. The 'Approvals' section shows 'Needs approval' by 'Frank Smith'.

If the field isn't showing, you may need to get a JIRA administrator to check the field is still available on your project screens.

How do my customers know who to add as an approver?

Make sure you add [descriptive titles and help text](#) for the approvals field on your portal. This should include details on who the customer should add (such as their manager, or a member of their IT team), and how many approvers. Depending on your [customer access settings](#), your customers may be able to select users or other customers from a drop-down list. Take this into account when adding your title and help text.

Setting up queues for your team

Make sure that your team is working on the right requests at the right time

with easily configurable service desk queues. A queue is a filtered set of issues that are displayed to your team. Use your project's default queues or create custom queues to save time triaging requests, and to give your agents more visibility of the number and type of incoming customer requests they need to work on.

On this page:

- [Creating new queues](#)
- [Managing queues](#)
- [How your team uses queues](#)

You can log in as a service desk administrator to configure queues on the aptly named **Queues** tab in your service desk project:

Creating new queues

When creating a new queue, you can select the queue name, the issues that will be filtered into this queue, and the columns that appear in the queue to make life easier for your service team. Here's how you create a new queue:

1. From your service desk project sidebar, select **Queues > New queue**.
2. Name your queue using language your team will understand (e.g. Issues due this week).
3. Select which issues will show up in this queue using the dropdown options in the Basic search view:

- You can also select the advanced search view to enter a [JIRA Query Language \(JQL\)](#) statement.
4. Add or remove columns to control what issue information, such as the issue key and issue creation date, is displayed in your queue.
 5. Select **Create**. If you have existing issues in your project that fit the criteria selected in "Issues to show", these issues will now appear in your new queue.

Managing queues

You can reorder or delete queues at any time by hovering over the Queues sidebar and selecting **Manage**. In the Manage queues dialog that appears, you can see the number of issues in each queue, and drag and drop queues to reorder them.

You can edit existing queues by selecting the queue you wish to configure and selecting **Edit queue** in the top right corner. You can edit the queue name, the issues shown, the columns, and column order. Note that you will see a live preview of the updated issues that appear in this queue as you configure it.

How your team uses queues

Queues give your agents a single view of the work that needs to be done across their team. Agents can view all of the queues in your service desk project; however, they cannot create or edit queues.

Setting up request types

JIRA Service Desk provides a set of default request types that are configured for basic IT help desk scenarios. You can configure the default request types or add new ones to suit the needs of your customers and team. Request types can be organized into groups to help customers find the request they need on the customer portal.

You must be a JIRA Service Desk Administrator or Project Administrator to configure request types and workflow statuses.

On this page:

- Set up request types
- Organize request types into groups
- Customize the fields on a request type
- Customize the workflow statuses for a request type
- Hidden fields and unsupported fields

Set up request types

Each request type in a service desk is based on an issue type. Open **Project settings > Request types** to manage your project's request types:

The screenshot shows the 'Request types' configuration page in JIRA Service Desk. It features a table of request types with columns for name, description, and actions. Annotations provide guidance on how to use the interface:

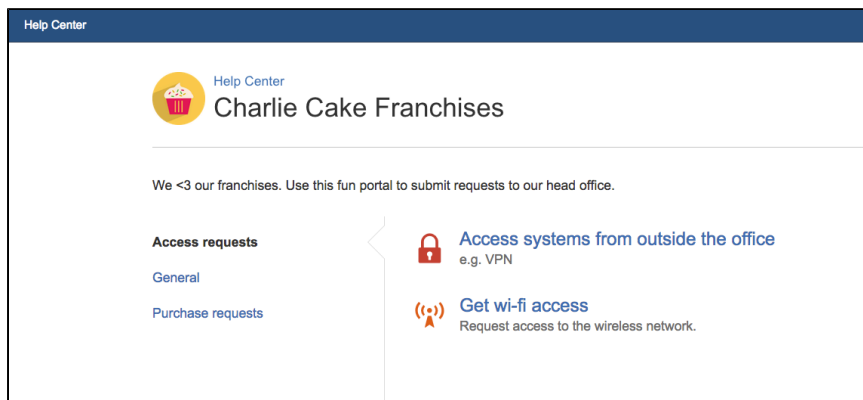
- Top left:** "Give the request an intuitive name by using keywords your customers look for." (points to the 'Report a system problem' name)
- Top center:** "Add a new request type by selecting the issue type the request is based on." (points to the 'Incident' issue type dropdown)
- Top right:** "Help your customers choose the correct request type by providing helpful descriptions." (points to the 'Having trouble with a system?' description)
- Bottom right:** "Customize request fields and workflow statuses, and add request types to groups on your customer portal." (points to the 'Edit fields' and 'Edit groups' links)

Name	Description	Issue Type	Actions
Report a system problem <small>EMAIL</small>	Incident	Having trouble with a system?	Edit fields Edit groups (3) x
Get IT help	Service Request	Get assistance for general IT	Edit fields Edit groups (2) x

A single issue type can be the basis for many different request types (for example, the "Purchase" issue type serves as the basis for both the "Request new hardware" and "Request new software" requests).

Organize request types into groups

We recommend using groups if you have seven or more request types, so you can make your request types easier to find on the customer portal. You must have more than one group for the groups to appear in the customer portal. For example: 'Access requests', 'General', and 'Purchase requests':

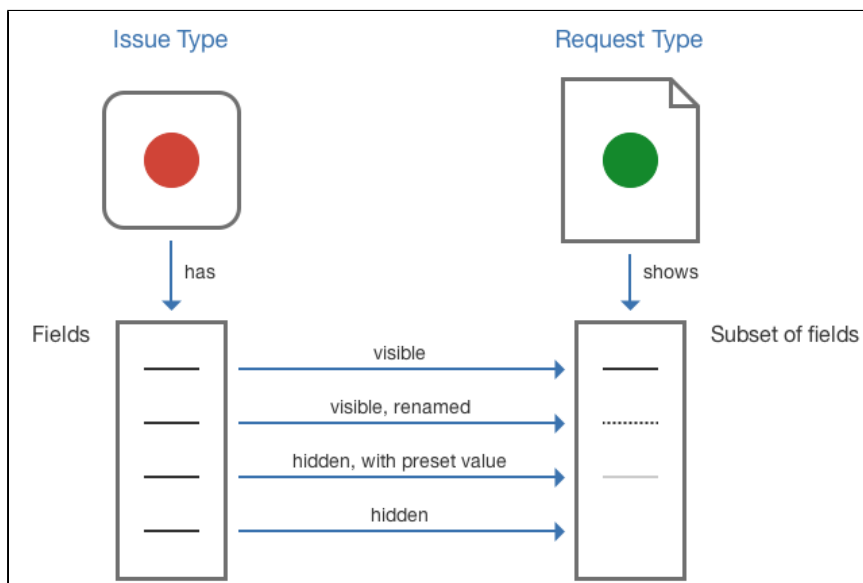


Administrators and project administrators can manage request type groups in **Project settings > Request types**. Click on a group to add new or existing request types to it. You can also create a new group by clicking **+Add group**. Note that request type groups are unique to a service desk project.

- You can drag and drop request types to re-arrange them in your groups (and, consequently, on your customer portal). Similarly, you can drag and drop groups to re-arrange them on your customer portal.
- If you assign multiple groups to a single request type, the request type will appear on multiple tabs.

Customize the fields on a request type

The fields and descriptions that appear in a request type are based on the field configured for the issue type (that is, the issue type the request type is based on).



When editing the request type fields, you can use the **Fields** tab to change the default JIRA field names to more customer friendly language. For example, the "Summary" field appears as "What do you need?" for customers.

Request types / Request new hardware

Fields Workflow Statuses

This request form is linked to the following issue type: **Purchase** [+ Add a field](#)

Order a new computer or piece of IT hardware [Links \[1\]](#)

Visible fields

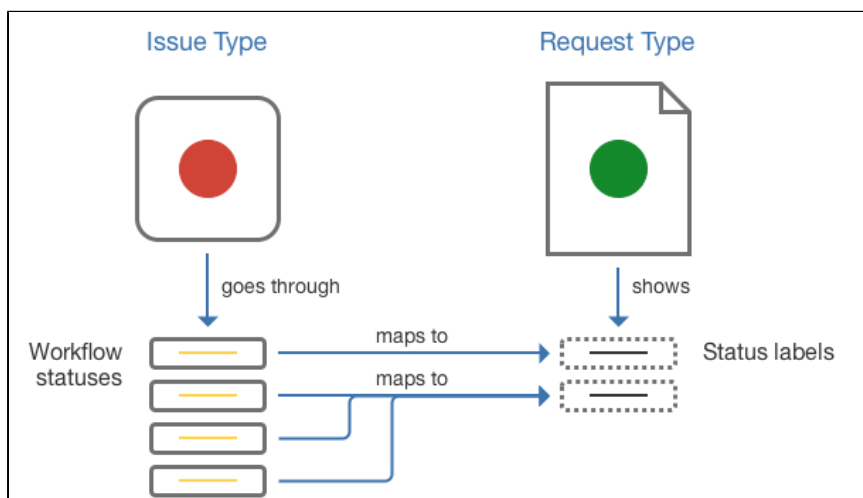
Display name	Required	Field help (Optional)
What do you need? Issue field: Summary	Yes	e.g. 'wireless PC keyboard'
Why do you need this?	Yes	
How urgent is it?	No	

You can also keep fields hidden but available on the request type so that their value can be used for other processes. For more details about how different types of fields work in JIRA Service Desk, see [Hidden fields and unsupported fields](#).

If the issue type doesn't have the fields you need, you must [add a field](#) to the JIRA issue type that the request type is based on. If the issue type uses multiple screen schemes, the new field must be available in the create screen. See [Associating a screen with an issue operation](#).

Customize the workflow statuses for a request type

JIRA Service Desk uses the workflow associated with the request's issue type for the flow of the request.



You can re-map the default workflow statuses to more customer friendly statuses that will appear for customers, and you can also map multiple statuses to a single customer status to simplify the appearance of the workflow. Use the **Workflow Statuses** tab to customize the workflow that customers will see.

Request types / Request new hardware

Fields Workflow Statuses

You can rename workflow statuses for this request type to make them more meaningful in the Customer Portal. If you set the same name for multiple statuses, these will appear as a single status. Customers will only be notified when there is a change in the status name.

Workflow status in JIRA	Status name to show customer
Resolved	Finished!
Waiting for Support	Waiting for 3rd party
Blocked Externally	Waiting for 3rd party
Waiting for Customer	Need info from requester
Untriaged	Waiting for Support

[Discard unsaved changes](#)

Only changes between these customer-visible 'status names' will be reflected in the Customer Portal and its notifications (e.g. a transition between two workflow statuses can be hidden on the Portal by giving them the same 'status name'). For more information about notifications, see [Configuring service desk notifications](#).

If you need to change the workflow of a request, you must edit the workflow associated with the service desk project by going to **Project settings > Workflow**.

Hidden fields and unsupported fields

Each request type in a service desk is based on an issue type. Every issue type has a set of allowed (and possibly required) fields associated with it. As you set up the request type, you can choose to include fields that are hidden on the customer portal but still provide a value to assist with your internal processes and reporting. For example, you might want to set the value of the "Label" field as "hardware" for the "Request new hardware" request type, and set the value as "software" for the "Request new software" request type.

Some fields used by an issue type are not supported for use in the customer portal; if you include these fields on a request type, they will automatically be added to the **Hidden fields with preset values** section and you'll be required to set a value for them.

Other fields aren't supported for use in JIRA Service Desk.

These fields can be added to the request type and given a preset value, but you cannot make them visible on the customer portal:

- Assignee
- Linked issues
- Any fields that are defined by other JIRA applications

These types of fields can't be added to a request type and won't appear in the "Add a field" dialog:

- Issue type
- Log work
- Reporter

- Security level
- Time tracking

Troubleshooting issues with request types

This page contains information about the errors and problems that you might have when setting up request types for your service desk.

Issue	Resolution
<p>Cannot delete the request type because it is the default request type for the email channel.</p> <p>▼ Details...</p> <p>If you see this error when trying to delete a request type, it means that the email channel for your service desk uses this request type as the default one for all the requests coming from emails. When JIRA Service Desk pulls emails from the associated email account and creates requests, this request type is assigned to the requests automatically.</p>	<p>JIRA administrators can change the default request type for email requests to be another one by going to Project settings > Email settings in your service desk. For more information about the email channel setup, see Receiving requests by email.</p>
<p>Cannot show a hidden field or make an optional field required because the request type is the default for the email channel.</p> <p>▼ Details...</p> <p>When JIRA Service Desk creates new requests from emails sent to the email account associated with your service desk, it copies the email subject to the Summary field and the email content to the Description field. When more fields are required, JIRA Service Desk cannot parse emails to fill them in with correct values.</p>	<p>You have the following options.</p> <ul style="list-style-type: none"> • If you want to show a hidden field, make it an optional one. • Ask your JIRA administrator to change the default request type for the email channel to use a different request type, and then modify your request type to include more required fields. You can also create a new request type for the email channel if no existing types are suitable. For more information about the email channel setup, see Receiving requests by email.

Request type displayed as "No match" in agent view.

▼ Details...

In JIRA Service Desk, the service desk request type is stored with the PortalKey/RequestTypeName value. For example, a "New Feature" request created in your "HelpDesk" Customer Portal would have the HelpDesk/NewFeature value. When you move this request to a new project, the HelpDesk/NewFeature value no longer matches the new project's Customer Portal name and request type values.

You have the following options.

- When you move a single issue to a new project, simply edit the service desk request type field with the correct request type:

- If you need to move a group of issues, you can search for issues with the same issue type in your existing project and then use the Bulk Edit wizard. In the third step, check **Change Customer Request Type** and select the request type that applies to this group of issues.

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

Setting up service desk users

The page introduces you to service desk project roles and groups, which you can use to manage service desk users and JIRA application licenses.

To set up and manage users, you must be logged in as an **administrator**. When you invite agents to a service desk project, these users are automatically added to the Service Desk Team project role, and assigned to groups associated with a service desk license. When you invite customers to a service desk project, these users do not consume a service desk license, but are similarly added to the Service Desk Customers project role and given restricted access to the customer portal.

On this page:

- [Overview of project roles](#)
- [Managing service desk agents](#)
- [Managing service desk customers](#)
- [How restricted customer access works](#)
- [How customers count towards license seats when using Atlassian Crowd](#)
- [Involving JIRA Software and JIRA Core users](#)
- [User management documentation](#)

Overview of project roles

The following table summarizes the default service desk project roles:

Project Role	Details
Administrators	<p>Users who administer a service desk project. In addition to what the Service Desk Team can do, Administrators can:</p> <ul style="list-style-type: none">• Add agents and customers to a service desk project• Remove agents and customers from a service desk• Configure request types, request security, and the email channel• Customize the customer portal• Create and edit reports and SLAs to track progress• Connect a Confluence knowledge base

Service Desk Team	<p>Users who work on customer requests. Agents assigned to this project role can:</p> <ul style="list-style-type: none"> • View queues, reports, SLA goals, and the customers list • Create and edit issues in the service desk project view and the customer portal • Add, edit and delete customer-facing and private comments on issues • Manage content in a connected knowledge base <p>Other JIRA application users can be assigned to the Service Desk Team role, but they will have limited project access.</p>
Service Desk Customers	<p>Users who create requests through the customer portal or by email. Customers can't log in to JIRA applications and don't have access to the service desk project view used by administrators and agents. Customers can:</p> <ul style="list-style-type: none"> • Create, comment on, and track requests through the customer portal • Create and comment on requests via email • Add comments and attachment to requests • Add other participants to their own requests (if the Request security settings allow it)

Managing service desk agents

Add agents to a service desk project

▼ [How to add agents...](#)

Easily invite new agents from your service desk project sidebar:

1. Select **Invite Team** and enter the agent's username (for existing system users) or email address (for new users). You'll see an updated JIRA Service Desk agent license count based on the number of new agents.
2. Select **Invite**.

Once invited, your new agents will receive an email with a link to your service desk project and be automatically added to the Service Desk Team project role and service-desk-users license group.

Remove agents from a service desk project

▼ [How to remove agents...](#)

To remove an agent from your service desk project:

1. In your project, go to **Project settings > Users and roles**.
2. Hover over the user or group you'd like to remove from the Service Desk Team project role and select



Note that removing an agent from your project does not revoke the agent's license. To free up an agent license, your administrator will need to proceed to



> **User management** and remove the agent's JIRA Service Desk application access.

Managing service desk customers

Add customers to a service desk project

You can manually invite customers, or [open your service desk](#) to allow customers to create their own service desk accounts.

▼ [How to add customers...](#)

To manually invite customers:

1. In your service desk project, select **Customers** and then select **Invite customers**.
2. Enter your customer's email address or invite multiple customers. Your customers will receive their new account details by email and will be added to the Service Desk Customer project role.

If your administrators have already enabled [public signup](#), you can open up your service desk project to allow new customers to create accounts on the customer portal or send requests by email, which creates an account for them automatically. To open up your service desk:

1. In your service desk project, go to **Project settings > Request security**.
2. Select "Anyone can sign up for a customer account" and select **Save**.

Remove customers from a service desk project

Removing customers from your service desk project depends on your project's request security settings, which you can check in **Project settings > Request security**.

▼ [How to remove customers...](#)

If your request security settings are set to "Only people on my customers list can raise a request", you can remove customers with these steps:

1. In your service desk project, go to **Project settings > Users and roles**.
2. Hover over the user or group you would like to remove from the Service Desk Customer role and select



If your request security settings are set to "Anyone can sign up for a customer account", or "Only people who have an account", your administrator will need to deactivate customer accounts by going to



> **User management**.

How restricted customer access works

The permissions assigned to customers are granted to the **Service Desk Customer - Portal Access** security type instead of the **Service Desk Customers** role. The **Service Desk Customer - Portal Access** security type gives people access to the Customer Portal only (not JIRA). This security type checks the **Service Desk Customers** role to determine who are customers. So in summary, the security type and the role work hand in hand to make sure that customers get the permissions they need to use the Customer Portal and cannot access JIRA.

For example, if you want your customers to be able to create requests through your Customer Portal, grant the **Create Issues** permission to the **Service Desk Customer - Portal Access** security type, not the **Service Desk Customers** role.

▼ [Why does the security type have more permissions than what customers can do?](#)

In the standard permission scheme, the **Service Desk Customer - Portal Access** security type has more permissions in place than the functionality available for customers to use. For example, the security type has the **Edit Own Comments** permission, but customers cannot do this on the Customer Portal. This is because JIRA Service Desk built the functions that we think are the most commonly used by service desk customers. We will evaluate the feature requests and expand the functions gradually. With the permissions in place now, future functionality additions to the Customer Portal will be easier because you will not have to modify permission schemes to make use of new functions in most cases. You can join the discussion on new features at our issue tracker: [0 issue](#).

How customers count towards license seats when using Atlassian Crowd

▼ [Click here to expand...](#)

This is usually caused by customers being a member of a group that has the JIRA users [global permission](#).

To prevent service desk customers from being counted toward your JIRA application license count:

1. Remove the customers from groups that have the permission.
2. If you have set default groups for new users to be added to in Crowd for the directory connected to JIRA, check if the groups have the **JIRA User** global permission assigned. If they do, you have two options:
 - Disable the default membership setting by removing the groups from the default group list. This means that your JIRA users (i.e. those that count towards your license) are not added

- to any group be default either when created.
- If you want JIRA users, JIRA Service Desk customers, or both to be added to default groups, you can do so by using separate directories for each type of user. Make sure that the default groups for JIRA Service Desk customers do not have the **JIRA Users** global permission.

Involving JIRA Software and JIRA Core users

JIRA application users without a JIRA Service Desk license (e.g. JIRA Software developers) can assist your team of agents on issues by working in the unlicensed view of a service desk project.

These users can:	<ul style="list-style-type: none"> View issues, comments and attachments Add and delete their own attachments Add and delete their own internal comments Watch and vote for issues
These users cannot:	<ul style="list-style-type: none"> See service desk queues, reports, and customers list Leave an external comment viewable by a service desk customer Log work on a service desk issue Transition a service desk issue Be assigned to a service desk issue

As an example of how to involve other JIRA application users, Martin, an IT service desk team agent, links an incident ticket in a service desk project to an underlying network problem ticket in a JIRA Software project. Andrew, a JIRA Software developer on the network operations team, assigns this network issue to himself and starts working on it. After fixing the problem, Andrew opens the linked service desk incident ticket and leaves an internal comment asking Martin to try the network connection again. After receiving the internal comment, Martin verifies the network connection and tells the customer that the problem is resolved.

Add JIRA Core or JIRA Software users to your service desk project

- In your project, go to **Project settings > Users and roles**.
- Select **Add users to a role**.
- Search for the users you'd like to add and choose the Service Desk Team role.
- Select **Add**.

JIRA Core and JIRA Software users in the Service Desk Team project roll will have restricted access to your service desk project and will only be able to see issues with which they're involved. Adding JIRA core and JIRA Software users to the Service Desk Team project role will not assign them to a service desk agent license.

Remove JIRA application users

- In your project, go to **Project settings > Users and roles**.
- To remove users from your service desk project only, simply hover over the user or group you wish to remove and select



To remove or deactivate users, your administrator will need to proceed to



> **User management**.

User management documentation

Check out the following documentation to learn more about managing users and permissions:

Documentation	Details
----------------------	----------------

User management	Learn more about adding and removing users, managing users with groups, and managing access to JIRA applications.
Managing project roles	Learn more about adding and removing project roles, and managing project role membership.
Enabling public signup	Open up your service desk project to allow customers without an account to sign up on the Customer Portal and send requests by email.
Configuring permissions	Learn how global permissions affect licensing, how project permissions are associated with a project role, and how to customize the default service desk project permission scheme.

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

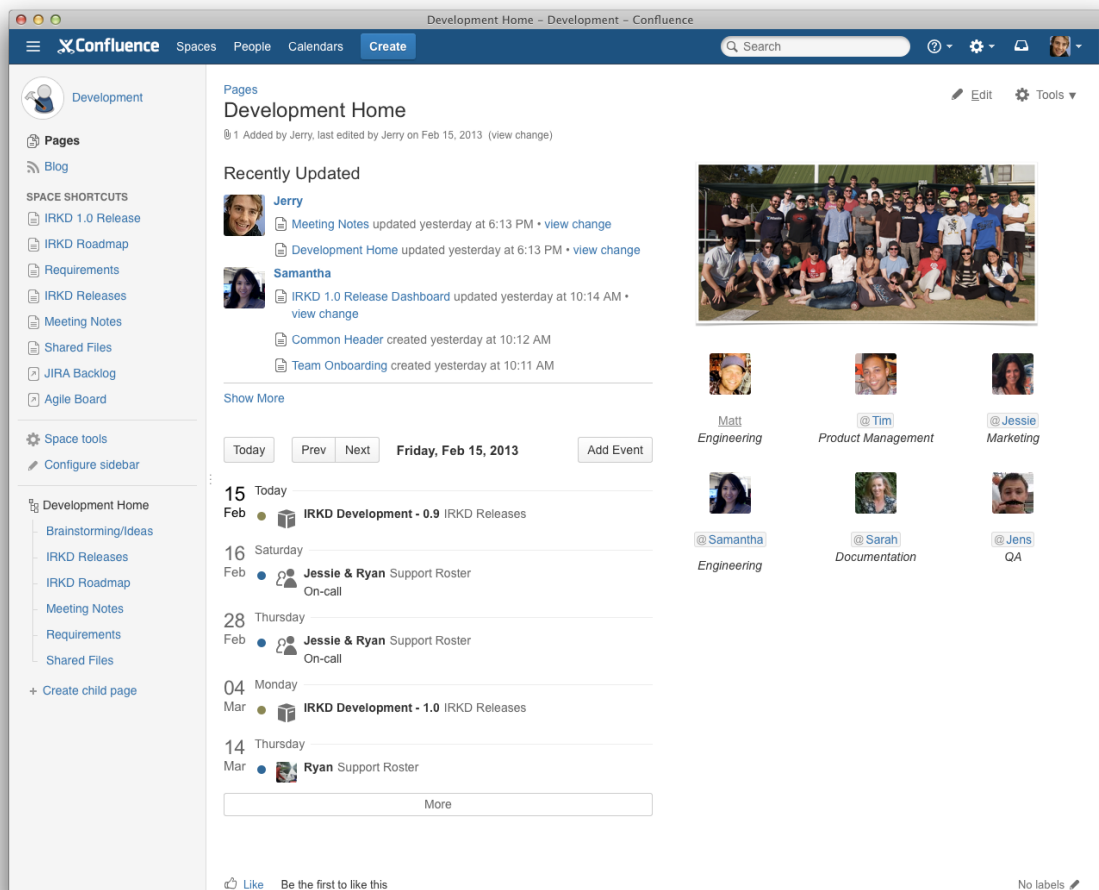


Since group membership can only be edited by users with the **JIRA Administrator** global permission, project administrators may therefore prefer to assign users, rather than groups, to their project roles.

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.



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

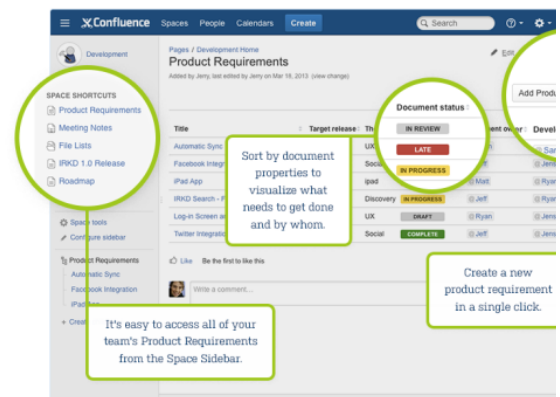
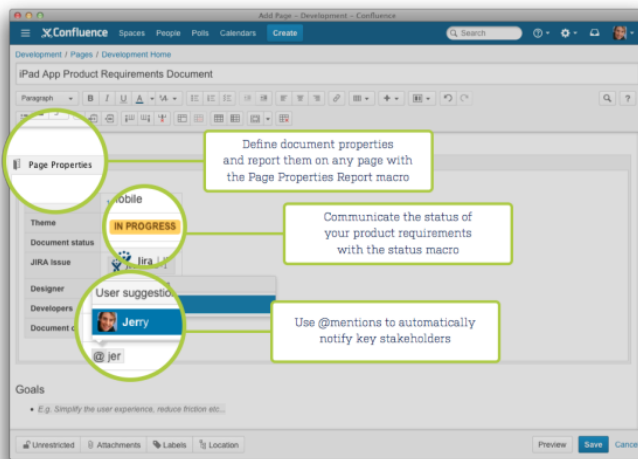
Confluence features for JIRA users

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.

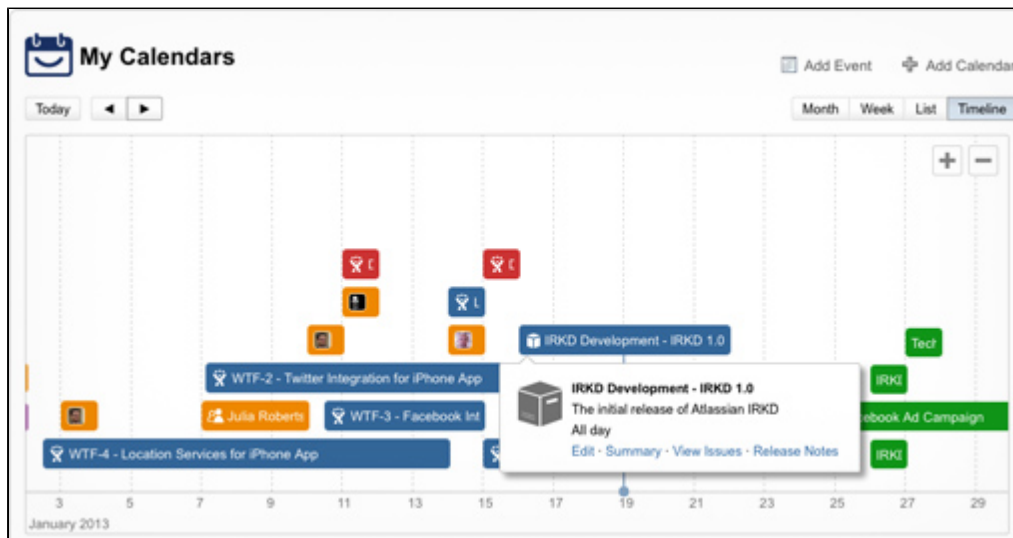
For more information, see [Blueprints](#).



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

confdev new JAC Search

Search using any issue key, search URL, JIRA link, JQL or plain text

Key	Summary
<input checked="" type="checkbox"/> CONFDEV-2362	Implement "What's New"
<input checked="" type="checkbox"/> CONFDEV-508	Add new shortcuts to RTE
<input checked="" type="checkbox"/> CONFDEV-3493	Existing table header row styling has been removed
<input checked="" type="checkbox"/> CONFDEV-1982	Migration has lost the table header styling
<input checked="" type="checkbox"/> 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 key, summary, type, created, updated, due, assignee, reporter, priority

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

Insert Cancel

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](#)

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

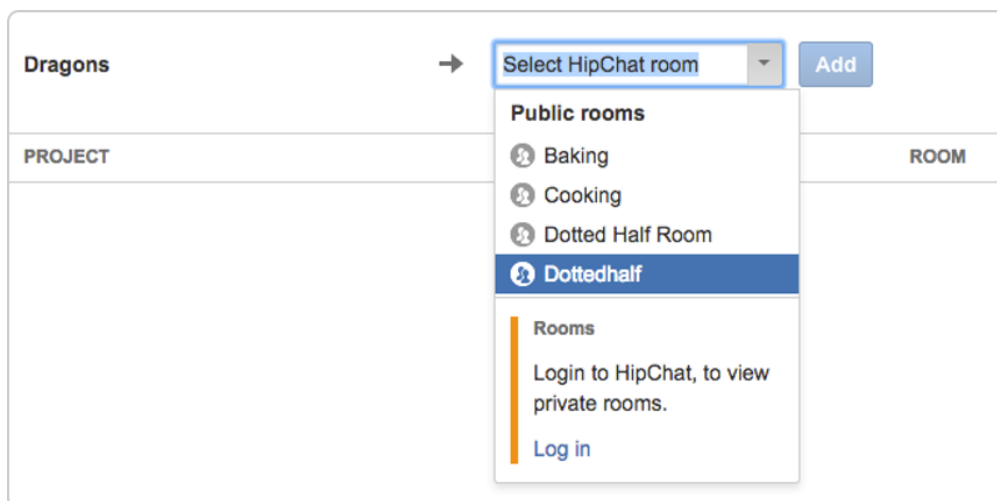
You can link JIRA projects with one or more HipChat rooms so that when issues are updated or created, messages are sent to the HipChat rooms that you specify.

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.

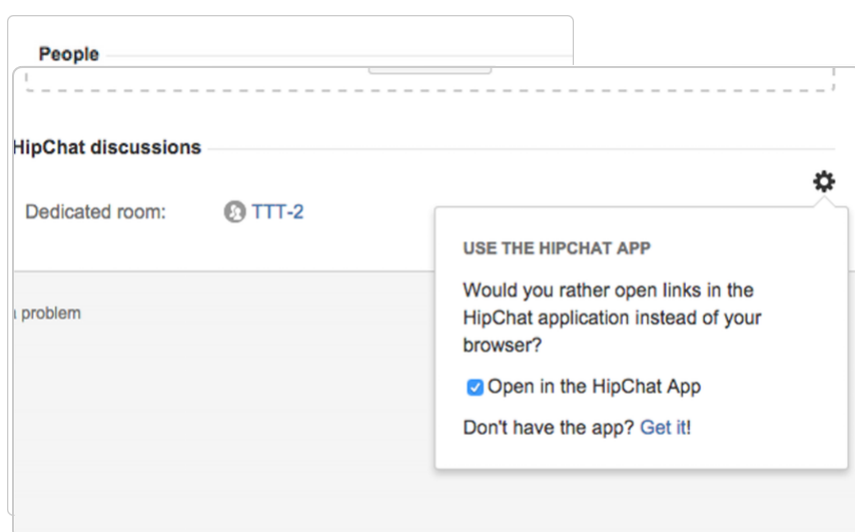
Invite users

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

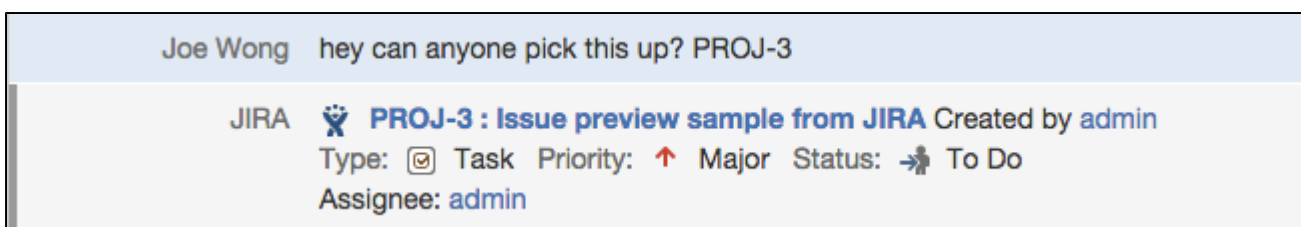
You can focus your discussion by creating or selecting a HipChat room to discuss an issue. When JIRA is integrated with HipChat and you are in the issue screen, you can select to "Create a room" or "Choose a room" in the **HipChat discussions** panel. This will associate the current issue and the room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



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**.

Default service desk project configuration

Use this page as a reference for the default configuration of your service desk projects, including custom fields, permissions, and database tables.

- [Custom fields](#)
- [Request types, issue types, and workflows](#)
- [Project permissions](#)
- [Security types](#)
- [Database tables](#)

Custom fields

If required, JIRA Service Desk will create the following custom fields:

Custom field	Type	Notes
Viewport Origin	String value, storing the 'Portal' and 'Request Type' if a request was created through the customer portal.	Issues must have this field to be a service desk request.
Time to resolution	An SLA field, stored in JSON format.	This field stores SLA information for time until a request's resolution is set. See Setting up SLAs for more information.
Customer Request Type	String value	Issues must have this field to be a service desk request.

Request types, issue types, and workflows

The default issue types, request types, and workflows are different for each service desk project type. When you create a new service desk project, you can view these defaults by selecting Request types, Issues types, or Workflows from the **Project settings** menu.

Project permissions

At installation time, JIRA Service Desk creates a project permission called **JIRA Service Desk agent access**. Users who require full access to service desk projects or functionality need to have this permission.

This page shows the permission configuration for a standard service desk project permission scheme.

- To see an overview of how permissions are set up for a service desk, see [Permissions overview](#).
- If you want to customize the permission scheme, see [Customizing JIRA Service Desk permissions](#).
- If you run into permission-related problems, see [Resolving JIRA Service Desk permission errors](#).

Security types

JIRA Service Desk introduces the **Service Desk Customer - Portal Access** security type. A security type is a concept that allows restriction of users to certain permissions, examples of security types include [project roles](#) and groups. **Service Desk Customer - Portal Access** is a special security type that only applies to users while they are viewing the customer portal; it was created specifically to allow customers to use the customer portal without giving them access to the internal service desk view and your other JIRA applications.

Database tables

When you set up JIRA Service Desk, the following tables will be created in your JIRA application database.

General JIRA Service Desk:

- AO_54307E_AGENTSIGNAUTRES
- AO_54307E_ASYNCUPGRADERECORD
- AO_54307E_CAPABILITY
- AO_54307E_CONFLUENCEKB
- AO_54307E_CONFLUENCEKBBENABLED
- AO_54307E_CONFLUENCEKBLABELS
- AO_54307E_CSATENTRIES
- AO_54307E_CUSTOMGLOBALTHEME
- AO_54307E_CUSTOMTHEME
- AO_54307E_EMAILCHANNELSETTING
- AO_54307E_EMAILSETTINGS
- AO_54307E_GOAL
- AO_54307E_GROUP
- AO_54307E_GROUPTOREQUESTTYPE
- AO_54307E_IMAGES

- AO_54307E_METRICCONDITION
- AO_54307E_PARTICIPANTSETTINGS
- AO_54307E_QUEUE
- AO_54307E_QUEUECOLUMN
- AO_54307E_REPORT
- AO_54307E_SERIES
- AO_54307E_SERVICEDESK
- AO_54307E_STATUSMAPPING
- AO_54307E_THRESHOLD
- AO_54307E_TIMEMETRIC
- AO_54307E_VIEWPORT
- AO_54307E_VIEWPORTFIELD
- AO_54307E_VIEWPORTFIELDVALUE
- AO_54307E_VIEWPORTFORM

JIRA Email Processor Plugin:

- AO_2C4E5C_MAILCHANNEL
- AO_2C4E5C_MAILCONNECTION
- AO_2C4E5C_MAILGLOBALHANDLER
- AO_2C4E5C_MAILHANDLER
- AO_2C4E5C_MAILITEM
- AO_2C4E5C_MAILITEMAUDIT
- AO_2C4E5C_MAILITEMCHUNK
- AO_2C4E5C_MAILRUNAUDIT

Automation:

- AO_9B2E3B_EXEC_RULE_MSG_ITEM
- AO_9B2E3B_IF_CONDITION_CONFIG
- AO_9B2E3B_IF_COND_CONF_DATA
- AO_9B2E3B_IF_COND_EXECUTION
- AO_9B2E3B_IF_EXECUTION
- AO_9B2E3B_IF_THEN
- AO_9B2E3B_IF_THEN_EXECUTION
- AO_9B2E3B_PROJECT_USER_CONTEXT
- AO_9B2E3B_RSETREV_PROJ_CONTEXT
- AO_9B2E3B_RSETREV_USER_CONTEXT
- AO_9B2E3B_RULE
- AO_9B2E3B_RULESET
- AO_9B2E3B_RULESET_REVISION
- AO_9B2E3B_RULE_EXECUTION
- AO_9B2E3B_THEN_ACTION_CONFIG
- AO_9B2E3B_THEN_ACT_CONF_DATA
- AO_9B2E3B_THEN_ACT_EXECUTION
- AO_9B2E3B_THEN_EXECUTION
- AO_9B2E3B_WHEN_HANDLER_CONFIG
- AO_9B2E3B_WHEN_HAND_CONF_DATA

JIRA Timed Promises Plugin:

- AO_F1B27B_HISTORY_RECORD
- AO_F1B27B_KEY_COMPONENT
- AO_F1B27B_KEY_COMP_HISTORY
- AO_F1B27B_PROMISE
- AO_F1B27B_PROMISE_HISTORY

Working on service desk projects

If you are an agent working on a JIRA Service Desk project, you're in the right

place!

- If this is the first time you have used JIRA Service Desk, check out [Getting started for service desk agents](#) for a brief introduction to your new workspace.
- If you're familiar with JIRA Service Desk, use the search bar below to find any needed information.

Search the topics in 'Working on service desk projects':

Working on issues

- [Working with issues](#)
- [Editing and collaborating on issues](#)
- [Attaching files and screenshots to issues](#)

Tracking your work

- [Keeping on top of SLAs](#)
- [Configuring dashboards](#)

Serving your customers

- [Raising requests on behalf of customers](#)

Using service desk queues

Customer requests become issues that you can view and work on in queues. JIRA Service Desk comes with default queues that your administrator can update to automatically triage issues for your team. As an agent, you can see how many issues are in each queue, and switch between queues to work on the right issues at the right time.

You can easily navigate to your service desk queues at any time by selecting **Queues** from your project sidebar:

The screenshot shows the JIRA Service Desk interface for the project 'Charlie Cake Franchises'. The left sidebar contains a 'Queues' section with a list of queues: 'Unassigned issues' (2), 'Assigned to me' (0), 'Waiting on customer' (0), and 'Recently resolved' (1). The main area displays the 'Unassigned issues' table with columns: Time to resolution, T, Key, Summary, Created, Reporter, and Due.

Time to resolution	T	Key	Summary	Created	Reporter	Due
3:51	🕒	CCF-2	New bakery printer	30/Sep/15	Administrator	
3:51	🕒	CCF-3	Need to access bakery wifi	30/Sep/15	Administrator	

At the bottom of the table, it shows '1-2 of 2'.

Switching queues

When you select **Queues** from your project sidebar for the first time, the secondary sidebar menu will open automatically. This sidebar displays all queues in your service desk project, as well as the number of issues in each queue. Simply select the name of the queue you wish to work from to view its issues.

To expand the view of a single queue, you can minimize your project sidebar by selecting



and minimize your queue sidebar by selecting



in the sidebar's upper right corner. When the queue sidebar is collapsed, a Switch queue dropdown will appear, which you can use to view a different queue or to reopen the queue sidebar:

Time to resolution	Created	Reporter	Due
3:45	30/Sep/15	Administrator	
3:46	30/Sep/15	Administrator	

Working with issues

In JIRA Service Desk, customer requests are automatically triaged into queues, so you can easily find the issues you need to work on. If you are ready to jump in and learn more about working on and managing customer issues, you're in the right place. 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.

On this page:

- [What is an issue?](#)
- [Next steps](#)

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.

In JIRA Service Desk, an issue is a packet of work that agents work on. In an IT service desk, it represents an incident, a change, and a service request, etc. For example, a customer request of "Our printer is not working" appears as follows in the customer portal:

Help Center / Charlie Cake Franchises

Our printer is not working **WAITING FOR SUPPORT**

Reference: CCF-4

People involved: Emma (Creator)

You can: [Add a comment](#), [Add attachment](#)

Details: Today 5:53 PM

Why do you need this?
I need to print handouts for a presentation

As an agent, you will pick the issue up internally in the service desk project to work on and it will look like the following:

Charlie Cake Franchises

CCF-4

Our printer is not working

Details: Type: IT Help, Priority: High, Status: WAITING FOR SUPPORT, Resolution: Unresolved

Description: I need to print handouts for a presentation

Attachments: Drop files to attach, or browse.

Activity: All, Comments, Work Log, History, Activity

SLAs: 4:00 Time to resolution within 4h

People: Assignee: Agent Will, Reporter: Emma, Request participants: None, Votes: 0, Watchers: 0

Service Desk request: Request type: Get IT help, Channel: JIRA

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](#)

Adding request participants

Customer requests are communications between the customer reporting an issue and the agent resolving that issue. To aid the resolution process, both agents and customers can add *request participants*. Request participants are additional JIRA Service Desk customers who can be included in the resolution process.

On this page:

- [Add other customers](#)
 - [Add request participants to an issue](#)
 - [Add request participants via email](#)
 - [Remove participants from an issue](#)
- [Add internal users](#)
- [About request participants](#)

Add other customers

Typically, you include other customers to ask them for more information or to update them about the issue. Request participants can add comments and attachments to a request, and receive the same notifications from JIRA Service Desk as the reporter. Participants can see who else is involved in a request both on the customer portal and in email notifications. This makes it possible for them to work from their inbox. They can also add more participants, for example, other customers who may be experiencing the same issue and would like to be notified about the resolution.

Add request participants to an issue

1. Navigate to an issue.
2. In the **People** section of the issue, add users to the **Request participants** field. You can only add existing customers to the service desk issue, regardless of how you configure the request security settings.

If customers need to add participants via the customer portal, they can do so by selecting **Share**. Service desk administrators can enable or disable this functionality by going to **Project settings > Request security**.

Add request participants via email

When you create or respond to a request via email, you can add request participants by adding their email address in the TO or CC fields.

Who you can add as a request participant via email depends on how you configure the [request security settings](#):

- If anyone can sign up for a customer account, then agents can add anyone as a request participant. If the person does not have an account in the service desk, then one is created for them.
- Customers can only add participants via email if the request security allows them to add participants to a request.

Remove participants from an issue

1. Navigate to an issue.
2. In the **People involved** section, click **Remove** under the participant's name.

Add internal users

As an example, you can involve other agents or JIRA Software developers on an issue to analyze a bug that a customer has reported. To involve internal users, you can mention them in a comment or add them as a watcher. As watchers, they will be notified about the issue and can communicate any updates to you internally.

About request participants

Request participants receive an email notifying them that they have been added. All customers, including the reporter, appear in the **People involved** section of the request both to the customer on the customer portal and to internal users in JIRA Service Desk.

Help Center / Office Administration R...

The printer won't connect to my computer

WAITING FOR SUPPORT

Comment on this request...

Reference: OAR-14

Details Today 11:09 AM

Description
I'm trying to print on level 8, but the printer isn't showing up on my list.

Priority
Major

People involved

Lily Williams
Creator

You can

[Add a comment](#)
[Add attachment](#)

Request participants
Lists participants on this request.

Note that involving people in a request does not bypass issue-level security. If issue-level security (e.g. restricting an issue to only be viewable by the reporter) has been applied, participants may not be able to access their requests. Service desk administrators can refer to the instructions in [Configuring Issue-level Security](#) to revise or delete an existing issue security scheme.

Attaching files and screenshots to issues

To share information with your customers, you can attach documents, images, and screenshots to your JIRA Service Desk issues. You can also restrict attachments to be viewed by your internal team only.

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 files and images to any issue in your service desk project. When working on an issue, simply drag and drop a file onto the issue, or select **More > Attach files**. You will then have the option to add a comment with more information about the attachment, and then share the file and comment with your customer or with your internal team only.

When adding or editing a comment, you can also select



to add attachments. In this case, you'll see wiki markup added to the comment field. As soon as you share your comment, you'll see the file preview.

▼ Acceptable file formats, characters, and sizes

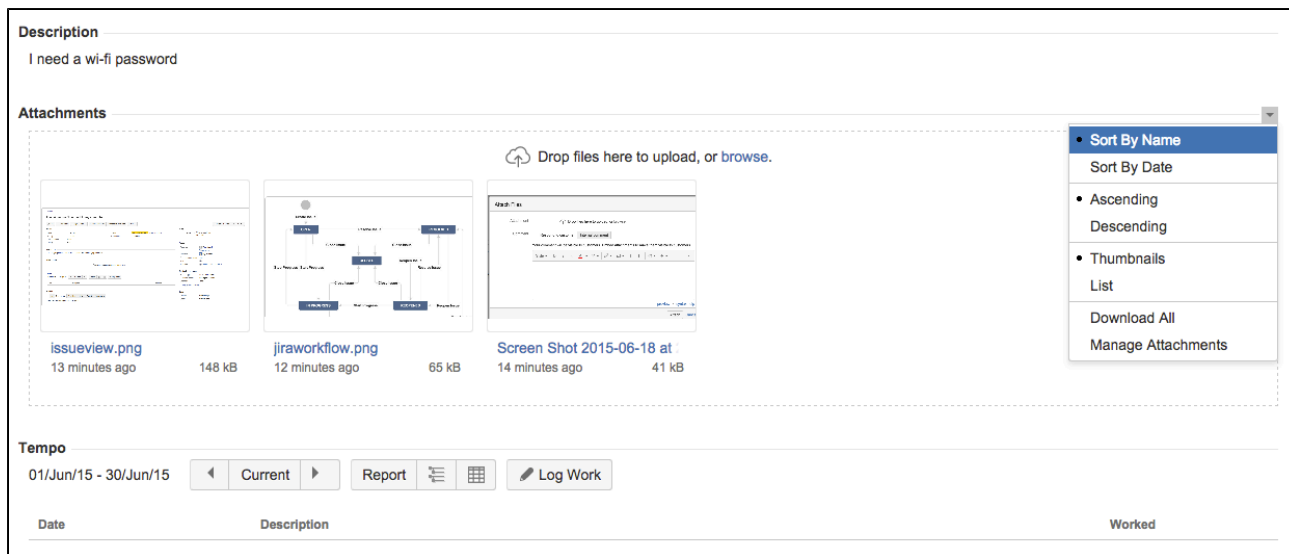
- File formats: GIFs, JPGs, PNGs
- A valid file name cannot contain any of these characters: '\', '/', '\"', '%', ':', '\$', '?', '*'.
- By default, the maximum size of any one file is 10MB, although this limit can be customized by your JIRA admin.

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



The selected criteria will be lost once you log out.



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.

1. Capture a screenshot using your system keyboard shortcut.
2. Paste the image from your clipboard onto the issue using your system keyboard shortcut or right-click menu. The **Attach screenshot** dialog will display.
3. Enter a filename.
4. Select **Upload**.

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.

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.

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.

i For Microsoft Excel and OpenOffice, it is not necessary to quote values in cells as these applications handle this automatically.

CSV file requirements

In addition to being 'well-formed', CSV files have the following requirements:

- **Each CSV file must possess a heading row with a Summary column**

The CSV file import wizard 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**, **Affect s 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.
- i** 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 (

JIRA-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 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**.

i 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	<p>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.</p> <p>You can also create new values on-the-fly during the import process.</p>
Labels	You can import issues with multiple labels by entering each label in a separate column.
Priority	<p>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.</p> <p>You can also create new values on-the-fly during the import process.</p>
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

Resolve your customer requests more efficiently with these tips and tricks for editing and collaborating on JIRA Service Desk 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 adding request participants
- 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.

In JIRA Service Desk, your administrator can also enable Request participants, which will appear as another issue field. You can add other agents and customers from your service desk project to help you resolve the original customer's request.

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.

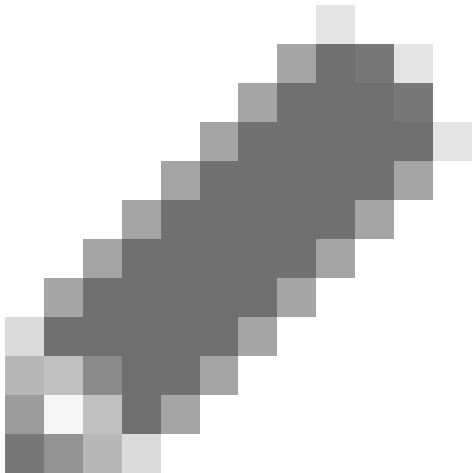




Commenting on issues

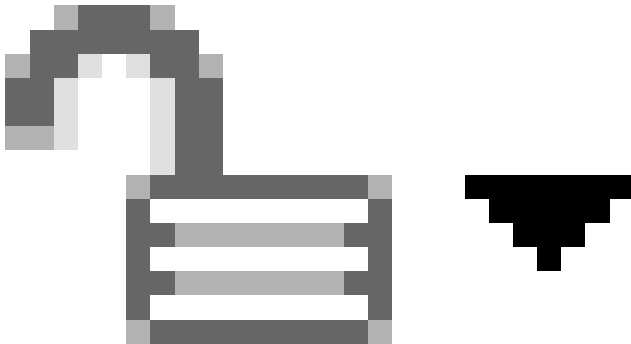

What permissions do you need?

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 select the internal (for other agents or collaborators) or external (for customers) tab.
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 data-bbox="303 1503 1315 1619"> <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 Permlink 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	17 May 2017 - Update documentation issue for JIRA 6.0	↑ ↓
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	14 May 2017 - Update final 5.2 docs to include webhooks "Exclude issue details" flag	↓ ↑
	JRADEV-16065	Update final 5.2 docs to include webhooks "Exclude issue details" flag	↓ ↑
	JRADEV-16150	14 May 2017 - Update final 5.2 docs to include webhooks "Exclude issue details" flag	↓ ↑

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.

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 * Charlie Cake Franchises (C...

Issue Type * Problem ⓘ
Some issue types are unavailable due to incompatible field configuration and/or workflow associations.

Created issue causes

Linked issues CCFA-3 × CCFA-4 × CCFA-5 × +
Search for issues to link to from the one you're creating.

Summary * Fix software bug in app XYZ

Description Style B I U A A Link Unlink Bulleted List Numbered List Table of Contents Attachments
 The print dialog is missing the Orientation features. The printer defaults to Portrait. Unable to print in Landscape orientation.

☒ Copy attachments
☒ Copy links

Create Cancel

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.
 - ⓘ 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.
 -  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.
 - 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.
 - Optional: In the second **Search** field, select the Confluence space to further narrow down the search.
 - 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

 **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.

 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 Service Desk to help your agents prioritize incoming customer requests and find overdue issues that need urgent attention. 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 these 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

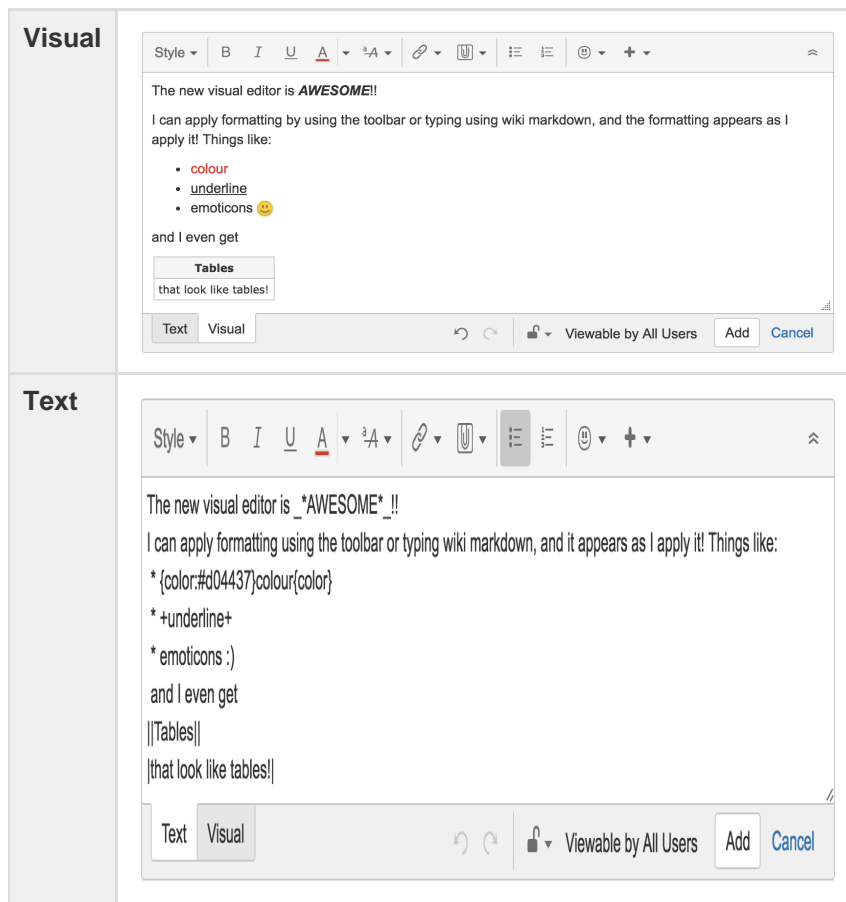
- Restricted comments appear to be removed after moving the issue. See this article: [Restricted comments disappear after moving an issue to a new project](#).

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.



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 it's 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.

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 Service Desk allows you to customize the configuration and behavior of issues to better suit the needs of your customers and agents. 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.

The image shows two side-by-side screenshots of the 'Create Issue' screen in JIRA, illustrating the difference between a default configuration and a customized one.

You can make this... (Left screen): This is the default 'Create Issue' screen. It features a 'Project' dropdown set to 'Development Strategy (DSE)', an 'Issue Type' dropdown set to 'Task', and a 'Field Tab' set to 'Group'. The 'Summary' field is required and is at the top. Below it are 'Account' (with a 'Please select' message), 'Due Date', 'Priority' (set to 'Medium'), and 'Environment'. The 'Description' field is at the bottom. There are 'Create another', 'Create', and 'Cancel' buttons at the bottom right.

...into this! (Right screen): This is a customized version of the 'Create Issue' screen. The 'Summary' field is now the first and largest field. Below it is the 'Environment' field, followed by the 'Description' field. The 'Project' and 'Issue Type' dropdowns are still present at the top. The 'Create another', 'Create', and 'Cancel' buttons are at the bottom right.

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.

Logging work on issues

In JIRA Service Desk, you use Service Level Agreements (SLAs) configured by your administrator to help you track how well you're meeting customer expectations (e.g. responding to a request within 4 hours). You can use the Time Tracking feature in addition to SLAs to generate a workload report when you're working on a customer request with other agents, or when you need to track time spent fixing a problem that affects multiple customer requests.

On this page:

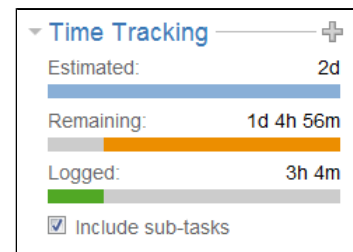
- [Before you begin](#)
- [Setting a time estimate for an issue](#)
- [Logging work on an issue](#)
- [Editing a work log entry](#)
- [Deleting a work log entry](#)
- [Customize d JIRA installations](#)

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.

- Open the issue and select **Edit**.
- 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.

- 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:

- Select the issue you want to log time on.
- Go to **More > Log Work**.
- 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	<p>Amount of time anticipated to resolve the issue after completing this unit of work. You can adjust this value using the following options:</p> <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.
Work description	<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.

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.


[Help Center](#) / [itservicedesk](#)

Please upgrade db.test.stg to PostGres 9.4
AWAITING APPROVAL

Your approval

Approve
Decline



Activity

Request requires approval. 1 approval needed. Today 10:34 AM LATEST

Details Today 10:34 AM

Description
 We need to upgrade our test staging DB to complete testing for our new environments.

 Who is your manager?
 Frank Smith

Reference: IT-4

You can
[Approve](#)
[Decline](#)
[Add a comment](#)
[Add attachment](#)

People involved
 Elaine Gould
 Creator

Awaiting approval
 Pending
 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.

Searching for issues

Can't find the customer issue you've been working on? This page will show you how to search for issues in JIRA Service Desk. Any agent can search for issues, although they will only see results from projects they have access to.

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

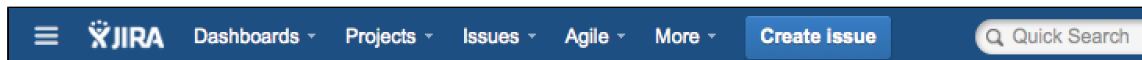
1. Define your search criteria

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

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

To use the quick search: Enter your search criteria in the search box in the header and press Enter.
Tip: If you know the issue key or project key, enter it before other search terms, e.g. "JIRA help lir

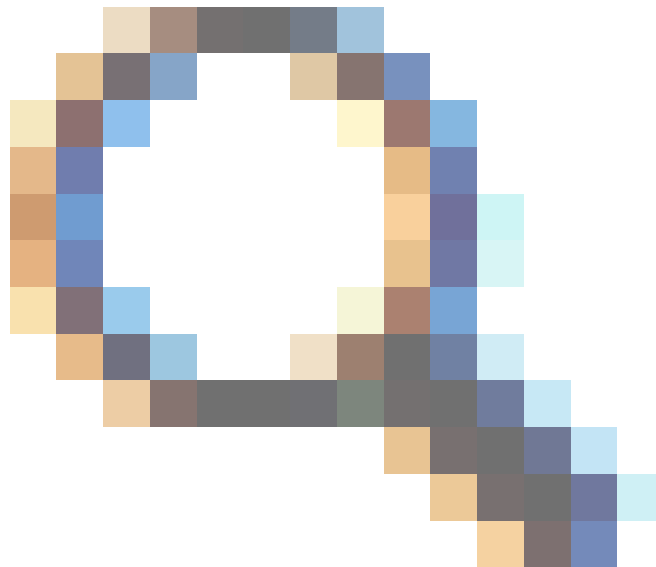


Basic search

The basic search is more precise than the quick search, but easier to use than the advanced search (a more user-friendly interface that lets you define complex queries, without needing to know how to use SQL or searching).

To use the basic search: Navigate to **Issues** (in header) > **Search for issues**, then enter your search criteria.

*Tip: If the advanced search is shown instead of the basic search, click **Basic** next to the*



icon.

Search

Sample Scrum ... ▾

Epic, Story ▾

In Progress, To... ▾

Assignee: All ▾

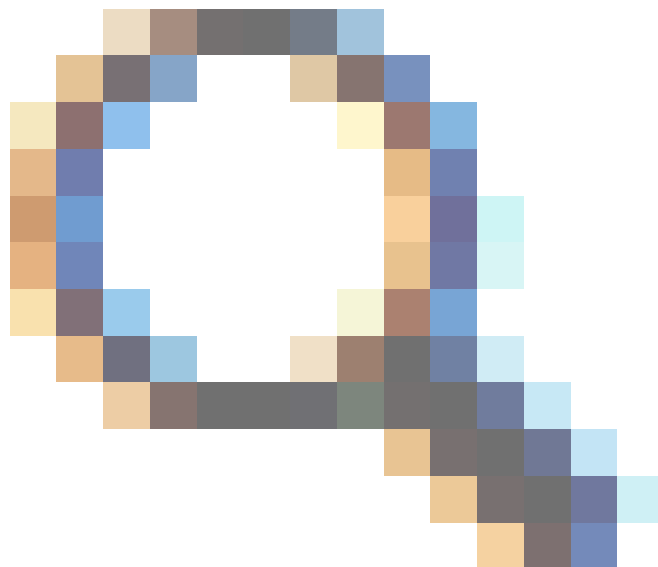
More ▾

Creator: Current User ▾

Advanced search

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 queries using the JIRA Query Language (JQL) to use this feature.

To use the advanced search: Navigate to **Issues** (in header) > **Search for issues**, then enter your search criteria. *Tip: If the basic search is shown instead of the advanced search, click **Advanced** next to the search icon.*



icon.

Search

Save as

Share

project = SSP AND issuetype in (Epic, Story) AND status in ("In Progress", "To Do") AND creator in (currentUser())

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.

Red Nerd											
<div> <div>1-8 of 8</div> <div> <div>Filters</div> <div>Undo</div> </div> <div> <div>Save as</div> <div>Details</div> <div>Star</div> </div> <div> <div>Share</div> <div>Export</div> <div>Tools</div> </div> </div>											
<div> <div>Find filters</div> <div>My Open Issues</div> <div>Reported by Me</div> <div>Recently Viewed</div> <div>All Issues</div> </div> <div> <div>FAVORITE FILTERS</div> <div>Browse Project Epics...</div> <div>Ignite docs</div> <div>Kickass docs</div> <div>PDL Needs Docs</div> <div>PDL-Page</div> <div>Red Nerd</div> <div>The Red Nerds need ...</div> </div>											
<div> <div>1-8 of 8</div> <div> <div>Text</div> <div>Basic</div> </div> </div>											
T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due	Fix Version/s
	ANGRY-304	Red Angry Nerd is scary	Unassigned	Bartosz Gatz	↓	Open	Unresolved	21/Mar/13	21/Mar/13		
	ANGRY-299	My red nerd is not working properly	Unassigned	Susan Griffin	↓	Open	Unresolved	15/Mar/13	15/Mar/13		
	ANGRY-158	ANGRY-13 / give the red nerd a yellow hat	Unassigned	Rosie Jameson	↑	Open	Unresolved	16/Apr/12	10/Aug/12		1.2
	ANGRY-13	I don't like the red nerd	Unassigned	Edwin Wong	↑	In Progress	Unresolved	27/May/11	25/Feb/13		1.2
	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	
	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
	ANGRY-79	Fix nerd's hair-styling	Bryan Rollins	Edwin Wong	↓	Closed	Fixed	12/Jul/11	20/Jan/12		1.3
	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 the basic search.

On this page:

- [Basic searching](#)
- [Running a saved search](#)
- [Troubleshooting](#)
- [Next steps](#)

Screenshot: Basic search

The screenshot shows the JIRA Basic Search interface. On the left, there's a 'FILTERS' sidebar with options like 'New filter', 'Find filters', 'My Open Issues', 'Reported by Me', 'Recently Viewed', and 'All Issues'. Below these are 'FAVORITE FILTERS' including '6.2-OD5 Docs', 'Enterprise issues ass...', 'JIRA documentation L...', 'm7 Ecosystem', and 'Open JRADEV issue...'. The main search area has a 'Search' button and a 'Save as' button. Below the search bar, there are search criteria: 'Sample Scrum', 'Epic, Story', 'In Progress, To...', 'Assignee: All', and 'Contains text'. There are also buttons for 'More', 'Advanced', and a search icon. The results section shows '1-4 of 4' items. The table has columns: Key, Summary, Assignee, Reporter, P, Status, Created, and Updated. The results are:

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

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:

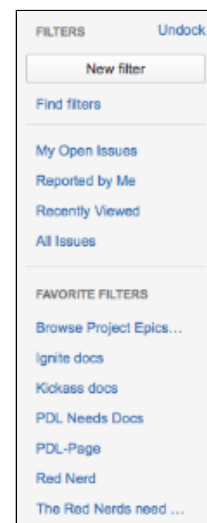
- the query contains an OR operator (note you can have an IN operator and it will be translated, e.g. `project in (A, B)`)
 - i.e. even though this query: `(project = JRA OR project = CONF)` is equivalent to this query: `(project in (JRA, CONF))`, only the second query will be translated.
 - the query 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 start 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

<code>r:</code>	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.
<code><project name></code> or <code><project key></code>	Find issues in a particular project.	<code>test project</code> <code>TST</code> <code>tst</code>
<code>overdue</code>	Find issues that were due before today.	<code>overdue</code>
<code>created:</code> <code>updated:</code> <code>due:</code>	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 <i>today</i> , <i>tomorrow</i> , <i>yesterday</i> , a single date range (e.g. '-1w'), or two date ranges (e.g. '-1w,1w'). Note that date ranges cannot have spaces in them. Valid date/time abbreviations are: 'w' (week), 'd' (day), 'h' (hour), 'm' (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.
<code><priority></code>	Find issues with a particular Priority.	<code>blocker</code> <code>major</code> <code>trivial</code>
<code><issue type></code>	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>
<code><resolution></code>	Find issues with a particular Resolution.	<code>fixed</code> <code>duplicate</code> <code>cannot reproduce</code>
<code>c:</code>	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:	Find issues with a particular Affects Version(s). To find all issues belonging to a 'major' version, use the wildcard symbol '*'.	<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.jspx?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

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due	Fix Versions
1	ANGRY-304	Red Angry Nerd is scary	Unassigned	Barlozz Gatz	+	Open	Unresolved	21/Mar/13	21/Mar/13		
2	ANGRY-299	My red nerd is not working properly	Unassigned	Susan Griffe	+	Open	Unresolved	15/Mar/13	15/Mar/13		
3	ANGRY-158	ANGRY-13 / give the red nerd a yellow hat	Unassigned	Rosie Jameson	+	Open	Unresolved	18/Apr/12	10/Aug/12		1.2
4	ANGRY-13	I don't like the red nerd	Unassigned	Edwin Wong	+	In Progress	Unresolved	27/May/11	25/Feb/13		1.2
5	ANGRY-306	Red Nerd should have his mouth open on impact	Unassigned	Barlozz Gatz	+	Open	Unresolved	21/Mar/13	05/Apr/13	31/Mar/13	
6	ANGRY-30	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
7	ANGRY-79	Fix nerd's hair-tying	Bryan Rollins	Edwin Wong	+	Closed	Fixed	12/Jul/11	20/Jun/12		1.3
8	ANGRY-70	Some graphical glitches	Christina Bang	Edwin Wong	+	In Progress	Unresolved	05/Jun/11	18/Sep/12		1.2

On this page:

- [Advanced searching](#)
- [Understanding advanced searching](#)
- [Reference](#)
- [Running a saved search](#)
- [Next steps](#)

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 = JIRA OR project = CONF)` is equivalent to this query: `(project in (JIRA, 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=JIRA`). 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
---------------	---	--

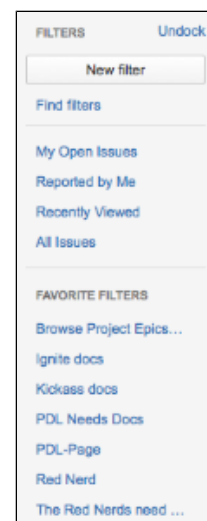
Operators	<p>An operator in JQL is one or more symbols or words that compare the value of a field on its left with one or more values (or functions) on its right, such that only true results are retrieved by the clause. Some operators may use the NOT keyword.</p>	<p>Operators reference page ▼ Show list 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 ▼ Show list of keywords</p> <ul style="list-style-type: none"> • AND • OR • NOT • EMPTY • NULL • ORDER BY

<p>Functions</p>	<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	affectedVersion
Field Type	VERSION
Auto-complete	Yes
Supported operators	= , != , > , >= , < , <= IS, IS NOT, IN, NOT IN <i>Not e 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	~ , !~ 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"> • releasedVersions() • latestReleasedVersion() • unreleasedVersions() • earliestUnreleasedVersion()

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	<ul style="list-style-type: none"> Find issues with an AffectedVersion of 3.14: affectedVersion = "3.14" <p><i>Note that full-stops are reserved characters and need to be surrounded by quote-marks.</i></p> <ul style="list-style-type: none"> 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	~ , != , !~ , > , >= , < , <= IS , IS NOT , IN , NOT IN , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED
Supported functions	<ul style="list-style-type: none"> approved() approver() myApproval() myPending() pending() pendingBy()
Examples	<ul style="list-style-type: none"> Find issues that require or required approval by John Smith: approval = approver(jsmith) Find issues that require approval by John Smith: approval = pendingBy(jsmith) Find issues that require approval by the current user: approval = myPending() Find all issues that require approval: approval = pending()

[^ 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	<p>= , !=</p> <p>IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</p> <p><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></p>
Unsupported operators	~ , !~ , > , >= , < , <=
Supported functions	<p>When used with the IN and NOT IN operators, this field supports:</p> <ul style="list-style-type: none"> membersOf() <p>When used with the EQUALS and NOT EQUALS operators, this field supports:</p> <ul style="list-style-type: none"> currentUser()
Examples	<ul style="list-style-type: none"> Find issues that are assigned to John Smith: <pre>assignee = "John Smith"</pre> or <pre>assignee = jsmith</pre> Find issues that are currently assigned, or were previously assigned, to John Smith: <pre>assignee WAS "John Smith"</pre> or <pre>assignee WAS jsmith</pre> Find issues that are assigned by the user with email address "bob@mycompany.com": <pre>assignee = "bob@mycompany.com"</pre> <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: <code>attachments IS NOT EMPTY</code> Search for issues that do not have attachments: <code>attachments IS EMPTY</code>
-----------------	---

[^ top of page](#)

Category

Search for issues that belong to projects in a particular category.

Syntax	<code>category</code>
Field Type	CATEGORY
Auto-complete	Yes
Supported operators	<code>=, !=</code> <code>IS, IS NOT, IN, NOT IN</code>
Unsupported operators	<code>~ , !~ , > , >= , < , <= WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues that belong to projects in the "Alphabet Projects" Category: <code>category = "Alphabet Projects"</code>

[^ top of page](#)

Comment

Search for issues that have a comment that contains particular text. [JIRA text-search syntax](#) can be used.

Syntax	<code>comment</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 comment contains text that matches "My PC is quite old" (i.e. a "fuzzy" match: <code>comment ~ "My PC is quite old"</code> Find issues where a comment contains the exact phrase "My PC is quite old": <code>comment ~ "\"My PC is quite old\""</code>

[^ 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	<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 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 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	<code>creator</code>
Field Type	USER
Auto-complete	Yes
Supported operators	<code>= , !=</code> <code>IS , IS NOT , IN , NOT IN , WAS , WAS IN , WAS NOT , WAS NOT IN</code>
Unsupported operators	<code>~ , !~ , > , >= , < , <=</code> <code>CHANGED</code>

Supported functions	<p>When used with the IN and NOT IN operators, this field supports:</p> <ul style="list-style-type: none"> membersOf() <p>When used with the EQUALS and NOT EQUALS operators, this field supports:</p> <ul style="list-style-type: none"> currentUser()
Examples	<ul style="list-style-type: none"> Search for issues that were created by Jill Jones: <pre>creator = "Jill Jones"</pre> or <pre>creator = "jjones"</pre> Search for issues that were created by the user with email address "bob@mycompany.com": <pre>creator = "bob@mycompany.com"</pre> <p><i>(Note that full-stops and "@" symbols are reserved characters, so the email address needs to be surrounded by quote-marks.)</i></p>

[^ 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	CustomFieldName
Alias	cf[CustomFieldID]
Field Type	<p><i>Depends on the custom field's configuration</i></p> <p><i>Note, JIRA text-search syntax can be used with custom fields of type 'Text'.</i></p>
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	<pre>= , != , > , >= , < , <=</pre> <pre>IS , IS NOT , IN , NOT IN</pre>
Unsupported operators: number and date fields	<pre>~ , !~</pre> <pre>WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</pre>

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"> • <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>
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"> • <code>releasedVersions()</code> • <code>latestReleasedVersion()</code> • <code>unreleasedVersions()</code> • <code>earliestUnreleasedVersion()</code>
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	<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> <div>Note that the Lucene value for Customer Request Type, is <code>portal-key/request-type-key</code>. 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.</div>
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	<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 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: <code>due < "2010/12/31"</code> Find all issues due on or before 31st December 2010: <code>due <= "2011/01/01"</code> Find all issues due tomorrow: <code>due = "1d"</code> Find all issues due in January 2011: <code>due >= "2011/01/01" and due <= "2011/01/31"</code> Find all issues due on 15 January 2011: <code>due = "2011/01/15"</code>
-----------------	--

[^ top of page](#)

Environment

Search for issues where the environment contains particular text. [JIRA text-search syntax](#) can be used.

Syntax	<code>environment</code>
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): <code>environment ~ "Third floor"</code> Find issues where the environment contains the exact phrase "Third floor": <code>environment ~ "\"Third floor\""</code>

[^ 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	<code>"epic link"</code>
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</code> , <code>savedFilter</code> , <code>searchRequest</code>
Field Type	Filter
Auto-complete	Yes
Supported operators	<code>=</code> , <code>!=</code> <code>IN</code> , <code>NOT IN</code>
Unsupported operators	<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>
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 jsmith: <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	<p><code>= , != , > , >= , < , <=</code> <code>IS , IS NOT , IN , NOT IN , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code></p> <p><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></p>
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>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 <code>characters</code>, 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	<p><code>= , != , > , >= , < , <=</code> <code>IS , IS NOT , IN , NOT IN</code></p>
Unsupported operators	<p><code>~ , !~</code> <code>WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code></p>

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	LABEL
Auto-complete	Yes
Supported operators	<p><code>= , !=, IS, IS NOT, IN, NOT IN</code></p> <p><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></p>
Unsupported operators	<p><code>~ , !~ , , > , >= , < , <=</code></p> <p><code>WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code></p>
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")</code> or <code>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	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 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	<ul style="list-style-type: none"> Find issues that are sub-tasks of issue TEST-1234: <code>parent = TEST-1234</code>
-----------------	--

[^ 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	<code>priority</code>
Field Type	PRIORITY
Auto-complete	Yes
Supported operators	<code>= , != , > , >= , < , <=</code> <code>IS , IS NOT, IN , NOT IN , WAS, WAS IN, WAS NOT, WAS NOT IN</code> <code>, CHANGED</code>
Unsupported operators	<code>~ , !~</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues with a priority of "High": <code>priority = High</code> Find issues with a priority ID of 10000: <code>priority = 10000</code>

[^ 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	<code>project</code>
Field Type	PROJECT
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	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	<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	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	<code>= , !=</code> <code>IS , IS NOT, IN , NOT IN , WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED</code>

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"> membersOf() <p>When used with the EQUALS and NOT EQUALS operators, this field supports:</p> <ul style="list-style-type: none"> currentUser()
Examples	<ul style="list-style-type: none"> Search for issues that were reported by Jill Jones: <pre>reporter = "Jill Jones" or reporter = jjones</pre> Search for issues that were reported by the user with email address "bob@mycompany.com": <pre>reporter = "bob@mycompany.com"</pre> <p>(Note that full-stops and "@" symbols are reserved <i>characters</i>, so the email address needs to be surrounded by quote-marks.)</p>

[^ 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> IS, IS NOT, IN, NOT IN
Unsupported operators	<code>~ , !~ , > , >= , < , <=</code> WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	<p>When used with the IN and NOT IN operators, this field supports:</p> <ul style="list-style-type: none"> email: requests submitted via email jira: requests created using JIRA portal: requests created using a Service Desk portal api: requests created using a REST API
Examples	<ul style="list-style-type: none"> Find issues where the request channel was email: <pre>request-channel-type = email</pre> Find issues where the request channel was something other than a service desk portal: <pre>request-channel-type != portal</pre>

[^ 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	<code>resolution</code>
Field Type	RESOLUTION
Auto-complete	Yes
Supported operators	<code>= , != , > , >= , < , <=</code> <code>IS , IS NOT , IN , NOT IN , WAS , WAS IN , WAS NOT , WAS NOT IN</code> <code> , CHANGED</code>
Unsupported operators	<code>~ , !~</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues with a resolution of "Cannot Reproduce" or "Won't Fix": <code>resolution in ("Cannot Reproduce" , "Won't Fix")</code> Find issues with a resolution ID of 5: <code>resolution = 5</code> Find issues that do not have a resolution: <code>resolution = unresolved</code>

[^ 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	<code>resolved</code>
Alias	<code>resolutionDate</code>
Field Type	DATE
Auto-complete	No
Supported operators	<code>= , != , > , >= , < , <=</code> <code>IS , IS NOT , IN , NOT IN</code>

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 that were resolved before 31st December 2010: resolved <= "2010/12/31" • Find all issues that were resolved before 2.00pm on 31st December 2010: resolved < "2010/12/31 14:00" • Find all issues that were resolved on or before 31st December 2010: resolved <= "2011/01/01" • Find issues that were resolved in January 2011: resolved > "2011/01/01" and resolved < "2011/02/01" • Find issues that were resolved on 15 January 2011: resolved > "2011/01/15" and resolved < "2011/01/16" • Find issues that were resolved in the last hour: resolved > -1h

[^ 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	sprint
Field Type	NUMBER
Auto-complete	Yes
Supported operators	= , != IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ , > , >= , < , <= WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED
Supported functions	<ul style="list-style-type: none"> • openSprints() • closedSprints()

Examples	<ul style="list-style-type: none"> Find issues that belong to sprint 999: <code>sprint = 999</code> Find issues that belong to sprint "February 1": <code>sprint = "February 1"</code> Find issues that belong to either "February 1", "February 2" or "February 3": <code>sprint in ("February 1", "February 2", "February 3")</code> Find issues that are assigned to a sprint: <code>sprint is not empty</code>
-----------------	--

[^ 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	<code>status</code>
Field Type	STATUS
Auto-complete	Yes
Supported operators	<code>= , !=</code> <code>IS , IS NOT, IN , NOT IN , WAS, WAS IN, WAS NOT, WAS NOT IN</code> <code>, CHANGED</code>
Unsupported operators	<code>~ , !~ , > , >= , < , <=</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues with a status of "Open": <code>status = Open</code> Find issues with a status ID of 1: <code>status = 1</code> Find issues that currently have, or previously had, a status of "Open": <code>status WAS Open</code>

[^ top of page](#)

Summary

Search for issues where the summary contains particular text. [JIRA text-search syntax](#) can be used.

Syntax	<code>summary</code>
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 summary contains text that matches "Error saving file" (i.e. a "fuzzy" match): summary ~ "Error saving file" Find issues where the summary contains the exact phrase "Error saving file": summary ~ "\"Error saving file\""

[^ 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	text
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 text field matches the word "Fred": text ~ "Fred" or text ~ Fred Find all issues where a text field contains the exact phrase "full screen": text ~ "\"full screen\""

[^ 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: timeSpent > 5d

[^ 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": type = Bug Find issues with an issue type of "Bug" or "Improvement": issueType in (Bug,Improvement) Find issues with an issue type ID of 2: issueType = 2

[^ 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	updatedAt
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	<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 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	<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"> 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: watcher = currentUser() Search for issues that the user "jsmith" is watching: watcher = "jsmith" Search for issues that are being watched by a member of the group "jira-administrators": watcher in membersOf("jira-administrators")

[^ 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: <code>watchers > 3</code>
-----------------	--

[^ 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	<code>workRatio</code>
Field Type	NUMBER
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	None
Examples	<ul style="list-style-type: none"> Find issues on which more than 75% of the original estimate has been spent: <code>workRatio > 75</code>

[^ top of page](#)

Advanced searching - keywords reference

This page describes information about keywords that are used for advanced searching. See [Advanced searching](#).

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

List of Keywords:

- AND
- OR
- NOT
- EMPTY
- NULL
- ORDER BY

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"
```

[^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

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

- 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 [JIRA 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.

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

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()

Examples

- Find all issues that are approved:
`approval = approved()`

- `startOfDay()`
- `startOfMonth()`
- `startOfWeek()`
- `startOfYear()`
- `subtaskIssueTypes()`
- `unreleasedVersions()`
- `votedIssues()`
- `watchedIssues()`

[^ top of page](#)

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	<code>approver(user,user)</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</code> <code> , CHANGED</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 or required approval by John Smith or Sarah Khan: <code>approval = approver(jsmith,skhan)</code>

[^ 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	<code>cascadeOption(parentOption)</code> <code>cascadeOption(parentOption,childOption)</code>
Supported fields	Custom fields of type 'Cascading Select'
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 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	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 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	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 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> <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 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	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 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)</code> or <code>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 <code>inc</code> 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	= , != , > , >= , < , <= WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED* <i>* Only in predicate</i>
Unsupported operators	~ , !~ IS , IS NOT , IN , NOT IN
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	<p>= , != , > , >= , < , <=</p> <p>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</p> <p>* Only in predicate</p>
Unsupported operators	~ , !~ 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	<p>= , != , > , >= , < , <=</p> <p>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</p> <p>* Only in predicate</p>
Unsupported operators	~ , !~ 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()</pre> <pre>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	<p><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></p>
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 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	<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 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	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 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	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	= , != , > , >= , < , <= WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED* <i>* Only in predicate</i>
Unsupported operators	~ , !~ IS , IS NOT , IN , NOT IN
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	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 the latest released version of the ABC project: <code>fixVersion = latestReleasedVersion(ABC)</code> Find issues that relate to the latest released version of the ABC project: <code>affectedVersion = latestReleasedVersion(ABC)</code> or <code>fixVersion = latestReleasedVersion(ABC)</code>
-----------------	--

[^ 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	<code>linkedIssues(issueKey)</code> <code>linkedIssues(issueKey,linkType)</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 that are linked to a particular issue: <code>issue in linkedIssues(ABC-123)</code> Find issues that are linked to a particular issue via a particular type of link: <code>issue in linkedIssues(ABC-123,"is duplicated by")</code>

[^ top of page](#)

membersOf()

Perform searches based on the members of a particular group.

Syntax	<code>membersOf(Group)</code>
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": <code>assignee in membersOf("jira-administrators")</code> Search through multiple groups and a specific user: <code>reporter in membersOf("jira-administrators") or reporter in membersOf("jira-core-users") or reporter=jsmith</code> Search for a particular group, but exclude a particular member or members: <code>assignee in membersOf(QA) and assignee not in ("John Smith","Jill Jones")</code> Exclude members of a particular group: <code>assignee not in membersOf(QA)</code>

[^ 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	<code>myApproval()</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</code> <code> , CHANGED</code>
Examples	<ul style="list-style-type: none"> Find all issues that require or have required my approval <code>approval = myApproval()</code>

[^ 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	<code>approved()</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</code> <code> , CHANGED</code>
Examples	<ul style="list-style-type: none"> Find all issues that require my approval <code>approval = myPending()</code>

[^ top of page](#)

now()

Perform searches based on the current time.

Syntax	<code>now()</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> <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 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: approval = pending(jsmith) Find issues that require by John Smith or Sarah Khan: approval = pending(jsmith,skhan)

[^ 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	<pre>projectsLeadByUser()</pre> <pre>projectsLeadByUser(username)</pre>
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: project in projectsLeadByUser() AND status = Open Find open issues in projects that are led by Bill: project in projectsLeadByUser(bill) AND status = Open

[^ 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	<pre>projectsWhereUserHasPermission(permission)</pre> <p>For the <code>permission</code> parameter, you can specify any of the permissions described on .</p>
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")</code> <code>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(rolename)</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 where you have the "Developers" role: <code>project in projectsWhereUserHasRole("Developers") AND</code> <code>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	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 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</code> <code>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	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 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. startOfDay("+1") is the same as startOfDay("+1d"). 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	= , != , > , >= , < , <= WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED* <i>* Only in predicate</i>
Unsupported operators	~ , !~ IS , IS NOT , IN , NOT IN
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	<p>= , != , > , >= , < , <=</p> <p>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</p> <p>* Only in predicate</p>
Unsupported operators	~ , !~ 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	<p>= , != , > , >= , < , <=</p> <p>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</p> <p>* Only in predicate</p>
Unsupported operators	~ , !~ 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	<pre>startOfYear()</pre> <pre>startOfYear("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	<p>= , != , > , >= , < , <=</p> <p>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</p> <p>* Only in predicate</p>
Unsupported operators	~ , !~ IS , IS NOT , IN , NOT IN
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	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 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	<pre>unreleasedVersions()</pre> <pre>unreleasedVersions(project)</pre>
---------------	--

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 `Windows95`, 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 [JIRA-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 `rooms`.

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 a 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)

The screenshot shows the JIRA issue page for 'Red Angry Nerd is scary' (ANGRY-304). The interface includes a top navigation bar with 'Save as' and 'Details' links. Below the navigation bar, there are filters for Project, Type, Status, Assignee, and a search bar. The left sidebar lists other issues under 'Order by'. The main content area displays the issue details, including Type (Bug), Priority (Minor), Status (Open), Resolution (Unresolved), and a description. An attachment is shown below the description. The right sidebar shows the 'People' section with assignee, reporter, and watchers, and the 'Dates' section with creation, update, and deployment dates.

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.

Manage Filters			
Favourite My Popular Search	My Filters ? Filters are issue searches that have been saved for re-use. This page shows all filters that you own.		
	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 ⚙
Powered by Atlassian Terms of Use Answers			

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

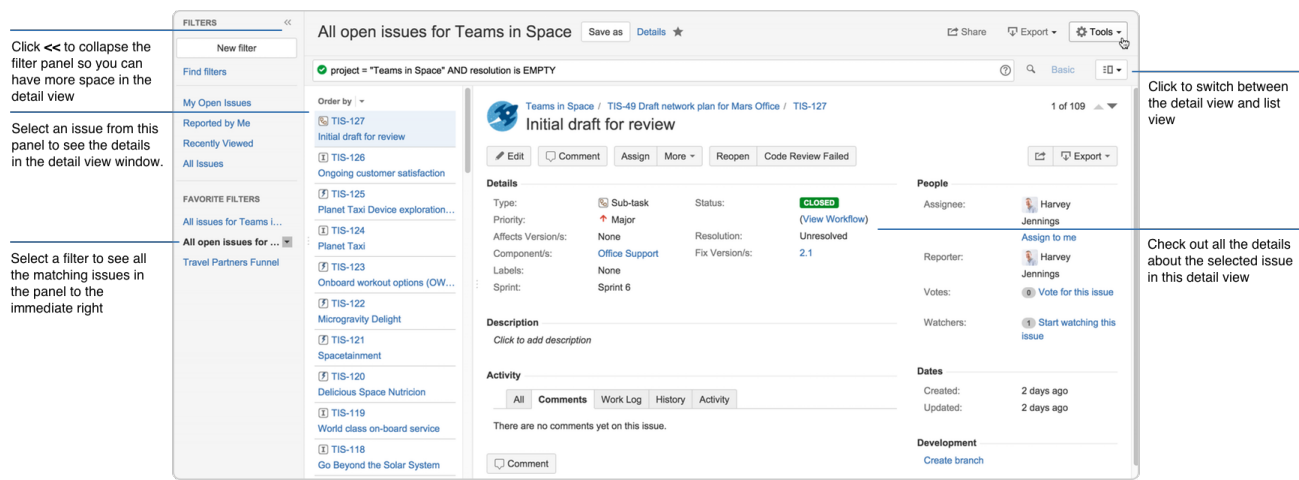
Once you have run a search, your search results will be displayed in the issue navigator. You may want to triage the entire list of issues or may be looking for just one. This page will show you what you can do with your search results, from changing what you see in the issue navigator to modifying the 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)



Changing your view of the search results

<p>List view or Detail view</p>	<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).
<p>Change the sort order</p>	<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.

Columns — show/hide and move	<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 in to 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.

View an issue	<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.
Action an issue	<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.

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	<code><link>https://extranet.atlassian.com:443/j:</code>
project (or pid)	<code><project id="10330" key="TST">Test</project></code>
description	<code><description>This is a detailed description</code>
environment	<code><environment>Sydney network</environment></code>
key	<code><key id="22574">TEST-4</key></code>
summary	<code><summary>This is a test</summary></code>
type (or issuetype)	<code><type id="3" iconUrl="https://extranet.atla</code>
parent	<code><parent id="22620">TEST-5</parent></code>
priority	<code><priority id="4" iconUrl="https://extranet.atlassian.com:44:</code>
status	<code><status id="5" iconUrl="https://extranet.atlassian.com:44:</code>
resolution	<code><resolution id="1">Fixed</resolution></code>

labels	<pre><labels> <label>focus</label> </labels></pre>
assignee	<pre><assignee username="jsmith">John Smith</as:</pre>
reporter	<pre><assignee username="jsmith">John Smith</as:</pre>
security	<pre><security id="10021">Private</security></pre>
created	<pre><created>Mon, 1 Sep 2008 17:30:03 -0500 (C</pre>
updated	<pre><updated>Mon, 1 Sep 2008 17:30:03 -0500 (C</pre>
resolved (or resolutiondate)	<pre><resolved>Mon, 1 Sep 2008 17:30:03 -0500 (C</pre>
due (or duedate)	<pre><due>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)>.</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" cr familiar</comment> <comment id="39273" author="jbrown" cr too</comment> </comments> </pre>
attachments (or attachment)	<pre> <attachments> <attachment id="30318" name="Issue Navig created="Mon, 9 Feb 2009 13:32:58 -0600 (C <attachment id="30323" name="Windows XP created="Tue, 10 Feb 2009 00:30:11 -0600 (C </attachments> </pre>
timeoriginalestimate	<pre> <timeoriginalestimate seconds="600">10 min </pre>
timeestimate	<pre> <timeestimate seconds="300">5 minutes</time </pre>
timespent	<pre> <timespent seconds="300">5 minutes</timespe </pre>
aggregatetimeoriginalestimate	<pre> <aggregatetimeoriginalestimate seconds="360 </pre>
aggregatetimeestimate	<pre> <aggregatetimerremainingestimate seconds="18000" </pre>
aggregatetimespent	<pre> <aggregatetimespent seconds="18000">5 hour; </pre>
timetracking	<pre> <timeoriginalestimate seconds="600">10 min <timeestimate seconds="300">5 minutes</time <timespent seconds="300">5 minutes</timespe <aggregatetimeoriginalestimate seconds="360 <aggregatetimerremainingestimate seconds="18000" <aggregatetimespent seconds="18000">5 hour; </pre>

issuelinks	<pre> <issuelinks> <issuelinktype id="10020"> <name>Duplicate</name> <inwardlinks description="is duplic <issuelink> <issuekey id="22477">INTSY: </issuelink> </inwardlinks> </issuelinktype> </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.custo <customfieldname>Department</custor <customfieldvalues> <customfieldvalue>Adminstratio </customfieldvalues> </customfield> </customfields> </pre>
allcustom	<pre> <customfields> <customfield id="customfield_10112" key="com.atlassian.jira.plugin.system.custo <customfieldname>Department</custor <customfieldvalues> <customfieldvalue>Adminstratio </customfieldvalues> </customfield> <customfield id="customfield_10111" key="com.atlassian.jira.plugin.system.custo <customfieldname>Expenditure Type<, <customfieldvalues> <customfieldvalue>Operating</ci </customfieldvalues> </customfield> </customfields> </pre>

Printable views

Printable	<p>Click Export > Printable.</p> <p>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.</p>
Full content	<p>Click Export > Full content.</p> <p>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.</p>

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	<p>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.</p> <ol style="list-style-type: none"> 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. <p>More information:</p> <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	<p>Click Export > RSS (Issues) or Export > RSS (Comments). The URL of the page that shows can be used in your feed reader.</p> <p>Tips:</p> <ul style="list-style-type: none"> • You can change the number of issues that are returned, by changing the value of the tempMax 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 os_authType=basic 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.

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 dashboards and permissions](#)
- [Sharing and editing your dashboard](#)
- [Adding favorite dashboards](#)
- [Note on dashboard permissions](#)
- [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.

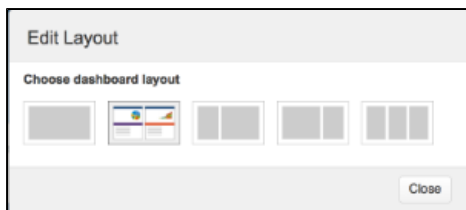
1. At the top right of the Dashboard, click the **Tools** menu.
2. Select either **Create Dashboard** to create a blank dashboard, or **Copy Dashboard** to create a copy of the dashboard you are currently viewing.
3. Name and describe your dashboard.
4. Fill out the rest of the fields as applicable.
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

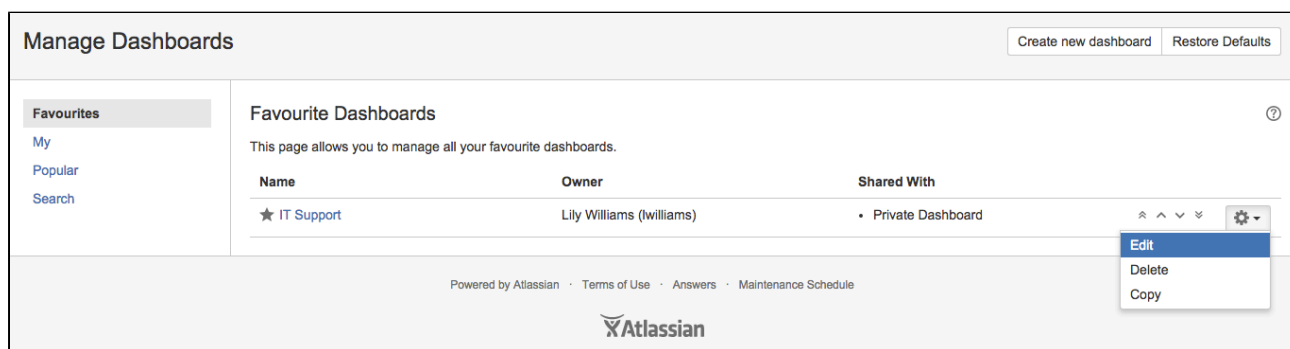


> **Edit/Share > Add sharing permissions.**

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.



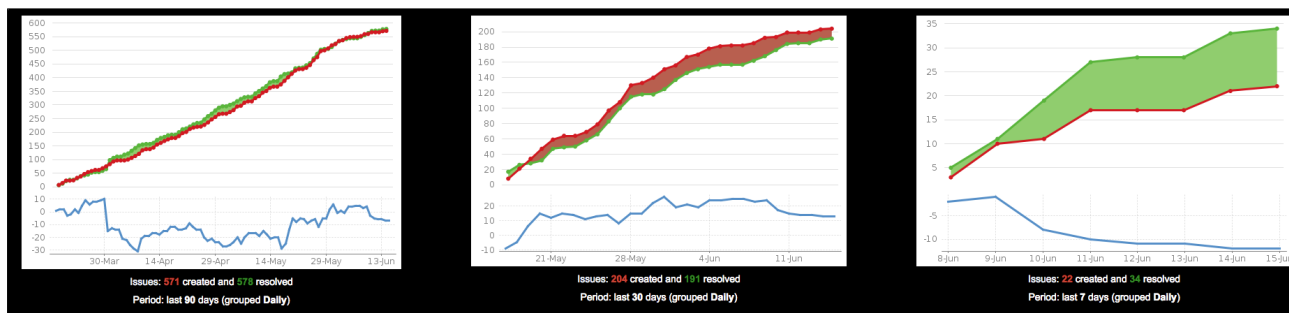
Note on dashboard permissions

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 Delete a gadget	Use the  button in the gadget header.

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.
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'.
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> <p><i>The website (container) you have placed this gadget on is unauthorized. Please contact your system administrator to have it approved.</i></p> </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 #ccc; padding-left: 10px; margin-left: 20px;"> <p><i>The website (container) you have placed this gadget on is unauthorized. Please contact your system administrator to have it approved.</i></p> </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; margin-left: 20px;"> <p><i>The website (container) you have placed this gadget on is unauthorized. Please contact your system administrator to have it approved.</i></p> </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: <div data-bbox="772 1541 1378 1637" style="border-left: 1px solid #ccc; padding-left: 10px; margin-left: 20px;"> <p><i>The website (container) you have placed this gadget on is unauthorized. Please contact your system administrator to have it approved.</i></p> </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>

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		✓		<p>Countdown! See how many working days you have before the current sprint ends.</p>
Favorite Filters	✓	✓	✓	<p>See a list of all the issue filters that have currently been added by you as a favorite filter.</p>
Filter Results	✓	✓	✓	<p>Seeing the results of a specified issue filter on the dashboard.</p>
FishEye Charts	✓	✓	✓	<p>Chart LOC data from a FishEye repository.</p>
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	✓	✓	✓	<p>See the issues returned from a specified project or saved filter (grouped by a specified field).</p>
Issues In Progress	✓	✓	✓	<p>Time to work! See all issues that are currently in progress and assigned to you.</p>

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	✓	✓	✓	<p>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.</p>
Labels Gadget	✓	✓	✓	<p>Use this gadget to see a list of all the labels used in a given project.</p>
Pie Chart	✓	✓	✓	<p>See the issues returned from a specified project or issue filter, grouped by a specified field.</p>
Quick Links	✓	✓	✓	<p>Link to frequently-used searches and operations.</p>
Recently Created Chart	✓	✓	✓	<p>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.</p>
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	✓	✓	✓	<p>View a list of test sessions.</p>

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 notifications](#)
- [Managing your user preferences](#)
- [Managing service desk preferences](#)
- [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.



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:

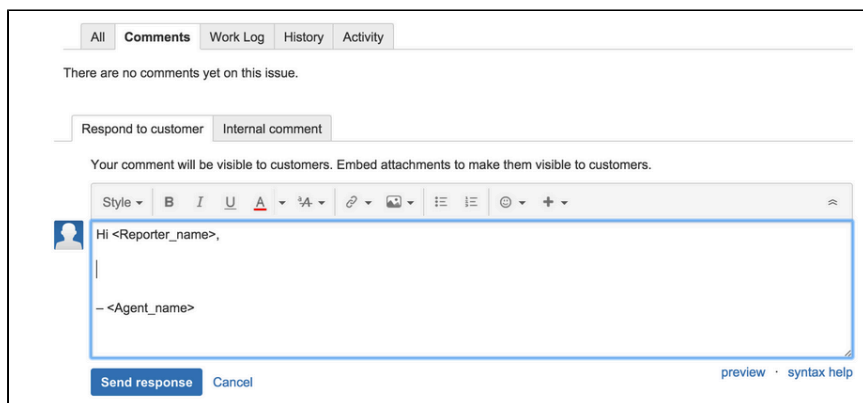
- The **Page Size**, or number of issues displayed on each Issue Navigator page
- Your preferred **language** from the drop-down list. If you don't see your preferred language in the list, see [Translating JIRA](#) for more information.
- Your **time zone** specified in your profile doesn't match the 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 'Comments' tab for an issue. Below the tabs, it says 'There are no comments yet on this issue.' There are two buttons: 'Respond to customer' and 'Internal comment'. Below these, a message states: 'Your comment will be visible to customers. Embed attachments to make them visible to customers.' A rich text editor is shown with a toolbar. The text area contains the pre-populated text: 'Hi <Reporter_name>,' followed by a line break and '— <Agent_name>'. At the bottom of the editor are 'Send response' and 'Cancel' buttons. To the right of the editor 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.

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.

On this page:

- [About OAuth access tokens](#)
- [Before you begin](#)
- [Issuing OAuth access tokens](#)
- [Revoking OAuth access tokens](#)

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](#).

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 Reflmp at http://localhost:8080/dashboards	02/10/2009	09/10/2009	Revoke OAuth Access Token
Created vs Resolved Chart	Atlassian Reflmp 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 page at http://localhost:8090 says:

If you revoke the access token, the application Activity Stream will no longer be able to access data using your account.


Hint: If this application accesses your data via a gadget, you can restore the permission later by clicking the lock icon on the gadget.

Click 'OK' to revoke the access token.

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	

Organizing work with components

Components are used to organize or group customer requests in a service desk project. You could set up a component for systems that your teams are responsible for (e.g. company intranet), and then set a default assignee so that any customer request about that system is assigned to the agent who manages it.

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 components 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.

Organizing work with versions

Versions are points-in-time for a project. They help you organize your work by giving you milestones to aim for. You can then assign the issues in your project to a specific version, and build up the work you need to do to complete that version.

On this page:

- Managing a project's versions
- Add a new version
- Release a version
- Archive a version
- Delete a version
- Merge multiple versions
- Reschedule a version

You need to have the project-specific **Administer Projects** [project permission](#) or the **JIRA Administrator** [global permission](#) to be able to:

- Add — create a new version against which issues can be aligned.
- Release — mark a version as released.
- Archive — hide an old version from the Releases report, and in the user interface.
- Delete — remove a version. You must choose an action for any issues with that version.
- Merge — combine multiple versions into one.
- Reschedule — re-arrange the order of versions.

Once a version has been created for a project, the 'Affects version' and 'Fix version' fields will become available for your issues. If you cannot see these fields on your issue, your project may not have any version yet, or the fields are hidden from view.

Managing a project's versions

The easiest way to manage a project's versions is through the Versions page.

1. Choose




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

2. Choose **Versions** in the sidebar. The **Versions** page is displayed, showing a list of versions.

Screenshot: The 'Versions' page

Name	Description	Start date	Release date
2.1	Bug fix and feature polish		27/Mar/13
3.0	5 New Levels, Android Support		17/Feb/13
2.0	UI cleanup		27/Mar/13
1.3	The Phantom Nerd		29/Jul/11
1.2	Return of the Nerds		15/Jun/11
1.1	The Empire Strikes Nerds		08/Jun/11
1.0	A New Hope for Nerds		01/Apr/11

Add a new version

1. The Add Version form is located at the top of the 'Versions' page.
2. Enter the name for the version. The name can be:
 - simple numeric, e.g. "2.1", or
 - complicated numeric, e.g. "2.1.3", or
 - a word, such as the project's internal code-name, e.g. "Memphis".
3. Optional details such as the version description (text not HTML), start date and release date (i.e. the planned release date for a version) can be also be specified. These can be changed later if required.
4. Click the **Add** button. You can drag the new version to a different position by hovering over the 'drag' icon
 
 at the left of the version name.

Release a version

1. On the 'Versions' page, hover over the relevant version to display the cog icon, then select **Release** from the drop-down menu.
2. If there are any issues set with this version as their 'Fix For' version, JIRA allows you to choose to change the 'Fix For' version if you wish. Otherwise, the operation will complete without modifying these issues.

To revert the release of a version, simply select **Unrelease** from the drop-down menu.

Archive a version

1. On the 'Versions' page, hover over the relevant version to display the cog icon, then select **Archive** from the drop-down menu.
2. The version list indicates the version 'archived' status with a semi-transparent icon. No further changes can be made to this version unless it is un-archived. Also it is not possible to remove any existing archived versions from an issue's affected and fix version fields or add any new archived versions.

To revert the archive of a version, simply select **Unarchive** from the drop-down menu.

Delete a version

1. On the 'Versions' page, hover over the relevant version to display the cog icon, then select **Delete** from the drop-down menu.
2. This will bring you to the 'Delete Version: <Version>' confirmation page. From here, you can specify the actions to be taken for issues associated with the version to be deleted. You can either associate these issues with another version, or simply remove references to the version to be deleted.

Merge multiple versions

Merging multiple versions allows you to move the issues from one or more versions to another version.

1. On the 'Versions' page, click the **Merge** link at the top right of the page.
2. The 'Merge Versions' popup will be displayed. On this page are two select lists — both listing all un-archived versions.
In the 'Merging From Versions' select list, choose the version(s) whose issues you wish to move. Versions selected on this list will be removed from the system. All issues associated with these versions will be updated to reflect the new version selected in the 'Merge To Version' select list. It is only possible to select one version to merge to.
3. Click the **Merge** button. If you are shown a confirmation page, click **Merge** again to complete the operation.

Reschedule a version

Rescheduling a version changes its place in the order of versions.

- On the 'Versions' page, click the



icon for the relevant version, and drag it to its new position in the version order.

Raising requests on behalf of customers

Let's say you're helping a customer resolve an issue over the phone, and you need to followup with additional information. You can use the Customer Portal to quickly enter your customer's name, fill in the issue details, and submit the service desk request.

If you don't need to create a request, but simply want to invite customers to your service desk so they know how to get help, you can skip ahead to [Invite a new customer](#).

On this page:

- [Raise a customer request](#)
- [Invite a new customer](#)

Note that your service desk administrator must have [public signup](#) enabled if you want to raise requests on behalf of new customers, or invite new customers.

Raise a customer request

1. From your service desk project sidebar, select **Customer Portal**.
2. Select the request type that matches your customer's need.
3. In the **Raise this request on behalf of** field, enter a new customer's email address or search for an existing customer's name:

4. Fill in the request details and select **Create**.

Your customer will be emailed a link to the new request – new customers will also receive an invitation to finish creating a service desk account – and you will be able to continue working on the issue from your service desk queue.

Invite a new customer

1. From your service desk project sidebar, select **Customers**.
2. Select **Invite customers** and enter your customer's email address.
3. Send your invitation and you're done!

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.

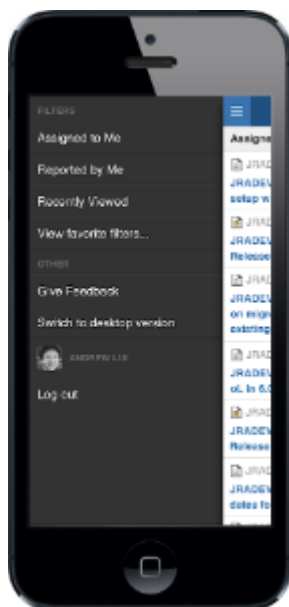
The JIRA mobile interface is designed for viewing and interacting with issues on the go. If you need full access

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.

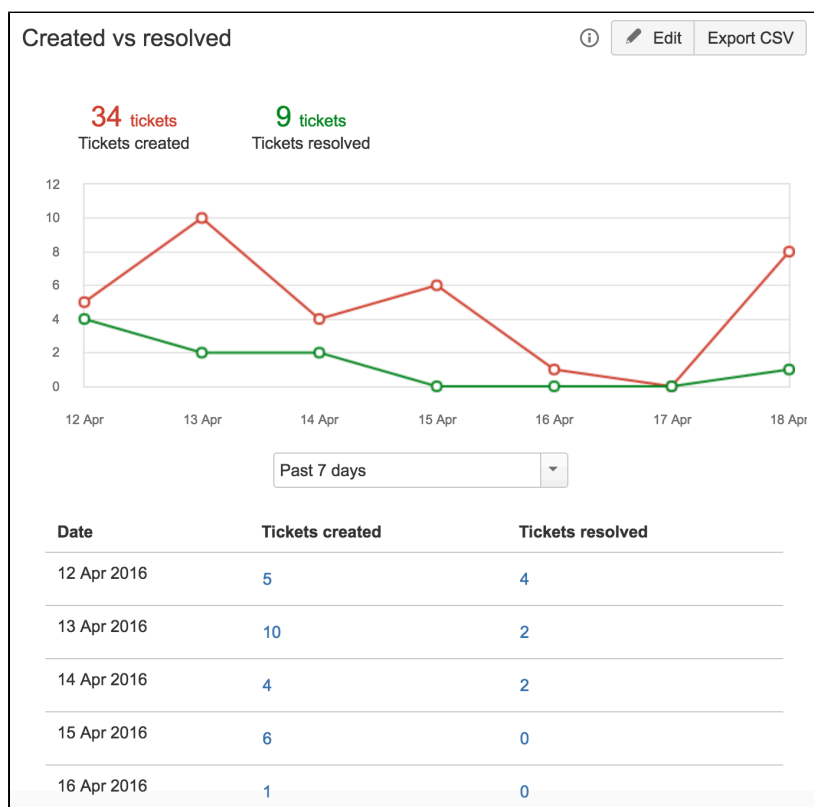
Can I disable JIRA mobile for my Cloud site?

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 an 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.

Setting up service desk reports

JIRA Service Desk provides powerful realtime reporting functionality so you can see your team's performance metrics:



On this page:

- [What makes up a service desk report](#)
- [Create a new report](#)

You can also create your own custom reports to query any combination of performance data. Your team members have access to a read-only version of the **Reports** tab so they can also see the data you're tracking. To create and edit reports, make sure you're logged in as an administrator.

What makes up a service desk report

Each report consists of one or more series. A report series is composed of a predefined series (e.g. issues created), or a time metric series (e.g. time to resolution). For each series, you need to select a label and color to easily identify the data points in this series on your report, and define which issues should be counted with a [JQL statement](#). When you create a service desk project for the first time, you'll see the following default reports depending on the project template you select:

Project template	Report groups	Default reports	Details
Basic Service Desk & IT Service Desk	Team	Workload	Shows how many requests your team is working on to help you ensure that your team's workload is evenly distributed

		SLA goals	Shows how your team is tracking towards each of the SLA goals you have set
		Satisfaction	Shows the average customer satisfaction rating for your team if you have opted to collect customer feedback
	Knowledge base	Article usage	Shows the number of times knowledge base articles were viewed and how many customers found them helpful.
		Article effectiveness	Compares the number of customers who were self-served through knowledge base articles to the number of customer who created a request.
	Custom	Created vs Resolved	Compares the number of issues that have been created by customers and resolved by your team in a specified time period
		Time to resolution	Compares the length of time taken to resolve requests of varying type or priority
		SLA met v breached	Compares the number of requests that have met or breached a selected SLA goal
Basic Service Desk only		Resolution by component	Compares the length of time taken to resolve requests for each component (e.g. office administration) in your service desk project
IT Service Desk only		Incident reports by priority	Compares the priority of incidents your customers have reported

Hovering over data points in a report will display the information for that specific data point.

Custom reports offer more detail beneath the report, and clicking on the data point allows you to drill down into the detail of the data point. You can export the contents of a custom report in a CSV format by selecting **Export CSV**. The CSV export will contain all the information presented below the report.

Create a new report

In your service desk project sidebar, select **Reports**. You will see a list of default reports for your service desk project. In this example, we'll create a report that shows the breakdown of all high priority issues based on the issue type:

1. Select **New report**
2. Choose a report name that you and your team will understand (e.g. High priority issues)
3. Select **Add a series** and fill in the following details:

Series = Created (to display all created issues that fit the series criteria)

Label = IT Help issues

Color =



JQL filter = type = "IT Help" AND priority = Highest OR priority = High

4. Select **Add** and save your report.
5. You can add additional series to capture high priority issues of other types (e.g. type = "Fault" or type = "Access")

See [Reporting on SLAs](#) for detailed information on how to run reports on SLA progress or status.

Setting up SLAs

JIRA Service Desk provides powerful built-in Service

Level Agreement (or SLA) management so you can track your team's progress against agreements you've set for customers. An SLA is made up of two settings:

- A time metric, which lets you define how time will be measured for this SLA; and,
- A goal for selected issues, which lets you define a target for the time metric. Different sets of issues can have different goals.

On this page:

- [Setting up SLA time metrics](#)
- [Setting up SLA goals](#)
- [Creating SLA Calendars](#)
- [How your team sees SLAs](#)
- [Managing SLA data](#)
- [SLA usage notes](#)

JIRA Service Desk comes with a few pre-configured SLA metrics to cover some of the most common IT requirements; however you can modify or create custom SLA metrics to reflect the SLAs you use in your business. JIRA Service Desk also provides robust reporting tools that you can use to track your team's performance against your SLAs. Check out [Reporting on SLAs](#) for tips on tracking your SLAs.

You must be logged in as a JIRA Administrator or Project Administrator to configure SLAs.

Setting up SLA time metrics

You can think of the time metric as a stopwatch that tracks time between two points in an issue's life-cycle. JIRA Service Desk lets you control exactly when time is tracked, letting you start, pause, and stop based on selected issue conditions. For example:

- In an SLA that guarantees issues are resolved in a certain amount of time, time starts when the issue is created and stops when the issue is resolved. You can choose to exclude time spent waiting on a customer by pausing the SLA on the Waiting for Customer status.
- In an SLA that guarantees a fast response time from your team, time starts when the issue status is Waiting for Support and stops when the status is Waiting for Customer. Each time the issue meets the start condition, a new cycle of the SLA will begin.

From your service desk project sidebar, select **Project settings > SLAs > New Metric** to fill in the following conditions:

Condition	Description
Metric name	This name (e.g. Time to resolution) will appear to agents in the SLA section of issues, and should indicate what you're tracking. Note that you can't change the name of an SLA metric after you've saved it.
Start	Time starts being counted against the SLA whenever the issue has this condition (e.g. Issue Created)
Pause on	Time doesn't get counted against the SLA whenever the issue has this condition (e.g. Status: Waiting for Customer)
Stop	Time stops being counted against the SLA whenever the issue has this condition (e.g. Resolution: Set)

Here's an example of the New Metric form:

Time to resolution

Save

Cancel

Time will be measured between the **Start** and **Stop** conditions below.

Start
Begin counting time when

Clear selected items

☒ Issue Created
 ☒ Resolution: Cleared
 ☐ Assignee: From Unassigned
 ☐ Assignee: To Unassigned
 ☐ Assignee: Changed
 ☐ Entered Status: Resolved
 ☐ Entered Status: Waiting for Cu...
 ☐ Entered Status: Waiting for Su...

→

Pause on
Time is not counted during

☐ Assignee: Set
 ☐ Assignee: Not Set
 ☐ Status: Resolved
 ☐ Status: Waiting for Customer
 ☐ Status: Waiting for Support
 ☐ Resolution: Set
 ☐ Resolution: Not Set

→

Stop
Finish counting time when

☒ Resolution: Set
 ☐ Assignee: From Unassigned
 ☐ Assignee: To Unassigned
 ☐ Assignee: Changed
 ☐ Entered Status: Resolved
 ☐ Entered Status: Waiting for Cu...
 ☐ Entered Status: Waiting for Su...
 ☐ Resolution: Cleared
 ☐ Resolution: Not Set

Notice that you can set multiple conditions for the start, stop, and pause time. Check out [Example: creating an SLA with multiple cycles](#) for an in-depth look at how you can use this functionality.

Setting up SLA goals

While the time conditions on an SLA specify what your team considers to be trackable time, the goal section of the SLA metric lets you set the amount of time that's allowed for different scenarios. SLA goals can be in hours or in time increments less than an hour. For example, an SLA that guarantees issues are resolved in a certain amount of time could have the following goals:

- Blocker issues are resolved within 24 hours and Critical issues are resolved within 36 hours
- Blocker issues created by a member of the Build Engineering team are resolved within 12 hours, while Blocker issues created by a member of the Accounting team are resolved within 36 hours.

In the New Metric or Edit Metric screen, you can fill out the following fields to define a goal:

Field	Description
Issues	You can enter specific issue criteria using JIRA Query Language (JQL) . Base goals on criteria that remain relatively constant throughout an issue's lifecycle (e.g. an issue's priority rather than its workflow status).
Goal	This is where you specify the target time for the SLA conditions you previously set. When specifying SLA goals that use a fraction of an hour, write the time as Xh Ym (e.g. 3h 30m). You can write SLA goals as hours and minutes, but not days.
Calendar	The calendar allows you to specify working hours when time can be counted against SLAs.

You can drag goals in order of importance. An issue is tracked against the first goal criteria it matches on this list:

Goals

Issues will be checked against this list, top to bottom, and assigned a time target based on the first matching JQL statement.

Issues (JQL)	Goal	Calendar	
<input type="text" value=""/> ?	<input type="text" value=""/> (e.g. 4h 30m)	24/7 Calendar (Default) ▾	<input type="button" value="Add"/>
priority = Critical	36h	24/7 Calendar (Default)	Delete
priority = Blocker	24h	24/7 Calendar (Default)	Delete
All remaining issues	48h	24/7 Calendar (Default)	

Creating SLA Calendars

By default, SLAs are measured against 24/7 working hours; however, you can use SLA Calendars to specify your team's working hours. For example, SLA calendars let you exclude lunch breaks, holidays, or weekends from the time that affects the SLA metrics.

Once you have saved your new SLA metric, select the **Calendars** button to add a calendar with your team's work schedule. The Sample 9-5 Calendar shown here can be edited to allow for a 1-hour lunch break that does not count towards the SLA goal. Simply delete the 09:00-17:00 line item and replace it with two line items (before and after lunch) for the same day:

Calendars

Sample 9-5 Calendar

Time zone

JIRA default (GMT-03:30) St Jo ... ▾

Work week

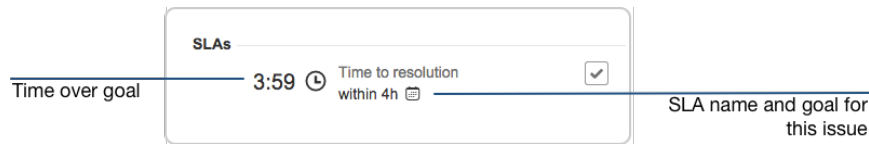
Monday ▾ 13 ▾ : 00 ▾ to 17 ▾ : 00 ▾

Monday	08:00	12:00	Delete
	13:00	17:00	Delete
Tuesday	09:00	17:00	Delete
Wednesday	09:00	17:00	Delete
Thursday	09:00	17:00	Delete
Friday	09:00	17:00	Delete

Save your calendar and open an SLA metric to associate the calendar with one of your metric goals. Note that SLA calendars are unique to each service desk project. See [Example: creating a basic SLA](#) for an example of setting up an SLA that uses a 9-5 working day SLA calendar.

How your team sees SLAs






Your team members can see a read-only version of the **SLA** tab so they can view how the SLA is configured. In the detail view of issues, the **SLA** section lists even more detail about the SLA that the issue is being measured against:



Review the following sections for more detail on what the SLA tracker conventions indicate.

Ongoing SLAs

The SLA tracker uses colors to indicate the urgency of a given SLA for an issue based on the time remaining.

	SLA has greater than 1 hour remaining.
	SLA has less than 1 hour remaining. If the SLA goal is one hour, the SLA has 30 minutes remaining.
	SLA has less than 30 minutes remaining. If the SLA goal is one hour, the SLA has 15 minutes remaining.
	SLA has breached the target. The amount of time past the goal is shown as a negative number.
	SLA has been paused.

Completed SLAs

A completed SLA displays the time remaining when the SLA was completed (or the amount of time breached) and an icon to indicate whether the SLA was completed successfully or unsuccessfully.

	SLA completed successfully.
	SLA completed unsuccessfully (it breached the target)

Multiple SLA targets

If the issue meets the criteria for multiple SLAs, trackers for each SLA will appear. In addition, if the SLA has had multiple cycles, you can hover over the symbols for more details on how the SLA was met for that particular cycle. (For example, in an SLA that is measured based on when an issue is waiting for support, you can see whether the SLA was met each time the issue started waiting for support.)

SLAs

0:44

72:41

0:59

Time waiting for support within 24 hours

Time to resolution PM within 96 hours

Time to response within 1 hour

Yesterday 5:36 PM - Yesterday 5:37

SLA sorting

When you view a list of issues (in a queue or elsewhere), you can sort them by their SLA resolution times. Ongoing issues are listed first, with the shortest time remaining at the beginning of the list. Completed issues are ranked last but aren't sorted by the remaining time.

Managing SLA data

When new SLA metric names are created, new custom fields are created in JIRA to store them. The type of these custom fields is SLA Custom Field Type. As a JIRA administrator, you have the following options to manage the SLA custom fields.

Allow project administrators to create SLA metric names

New metric names create new custom fields. You can restrict the creation of them to only be available for JIRA administrators.

1. Choose



> **Applications**. Scroll down to the **JIRA Service Desk** section and choose **Configuration**.

2. Use the **Allow project administrators to create SLA custom fields** option. If the setting is disabled, service desk administrators can only select from existing metric names when creating SLAs.

Clean up unused custom fields

You can find out if there are SLA custom fields that are not used by any SLA metrics and clean them up with one simple click. On the configuration page, the **Number of SLA fields currently not in use** menu shows the number of unused custom fields, if any. To delete them, click the **Clean up** button.

SLA usage notes

- Having the same user assigned to both the reporter and assignee roles may cause your SLA to work incorrectly.
- If you edit an existing SLA, JIRA Service Desk will re-index all the existing issues in the project; the re-indexing will ensure that the SLA status on the open issues reflects any changed criteria. All the historical SLA data for elapsed time will be recalculated to measure against the new metrics. Note that the SLA status is only recalculated for open issues and not for resolved issues.
For example, when the goal for Blocker issues changes from 6 hours to 4 hours, all the closed issues are still considered having met the goal as long as they were resolved in less than 6 hours. This ensures that your reports on closed issues remain accurate for the issues' lifecycle.
- If issue data changes in such a way that the goals for the issue change (for example, the priority changes from Critical to Blocker), the time against the previous goal will be tracked against the new goal. In other words, if the support team spent an hour on a Critical issue, when the issue is escalated to Blocker, the hour still counts against the new goal, even if it causes the SLA to be breached.
- Setting up a goal to be dependent on a different SLA is not recommended.

Reporting on SLAs

JIRA Service Desk provides robust [reporting tools](#) that you can use to track your team's performance against your SLAs. This page lists the SLA-specific JQL conditions you can use to query the SLA data in your service desk, as well as examples for creating some common JQL queries on SLAs.

- [State conditions](#)
- [Duration conditions](#)
- [Common SLA queries](#)

State conditions

State conditions are JQL functions used with operators = or != . For example:

```
"Time to resolution" = breached() or "Time to resolution" != breached()
```

Success/fail functions

Function name	with =	with !=
---------------	--------	---------

breached()	Gives all issues whose SLA last cycle (completed or ongoing) has breached (target goal failed)	Gives all issues whose SLA last cycle has not breached (for completed) or not breached yet (for ongoing cycles)
everBreached()	Gives all issues whose SLA has any cycle (current or past) that has ever breached.	Gives all issues whose SLA has all cycles (past or present) successful or not breached yet (if ongoing).

SLA state functions

This state addresses the last SLA cycle. This cycle can be completed (the stop event is reached) or ongoing (the stop event is not reached yet). When the cycle is ongoing, the cycle can be running or paused (if pause condition is true).

SLAs that have no cycles yet (the cycle has never been started) are not returned by these conditions.

Function name	with =	with !=
completed()	Gives all issues whose SLA last cycle is completed	Gives all issues whose SLA last cycle is not completed
running()	Gives all issues whose SLA last cycle is ongoing and not paused	Gives all issues whose SLA last cycle is not running (i.e. completed or paused)
paused()	Gives all issues whose SLA last cycle is ongoing and paused	Gives all issues whose SLA last cycle is not paused (i.e. completed or running)

Duration conditions

Conditions on duration are JQL functions used with operators <, <=, >, >=.

The '=' and '!=' operators are not supported.

These functions only apply to SLAs whose last cycle is ongoing (running or paused). Completed SLAs or SLAs without cycles will not be returned.

Example:

```
"Time to resolution" < elapsed(2h) or "Time to resolution" <
remaining("2h 30m")
```

There are two duration conditions:

Function name	Description
elapsed()	Gives issues whose SLA last cycle match condition on elapsed time since start event.

remaining()	<p>This function gives issues whose SLA last cycle match condition on remaining time before SLA breaches current goal target duration.</p> <p>This function is implicit, meaning that</p> <pre>"Time to resolution" > 5h</pre> <p>is the same as</p> <pre>"Time to resolution" > remaining(5h)</pre>
-------------	--

Common SLA queries

This table lists some examples of common SLA queries; the conditions you use for your own reports will vary depending on the way your JIRA project is set up.

To find out	Query
All issues that are about to break SLAs	"Time to first response" < 1h and "Time to first response" != breached()
Issues that have plenty of time until they are due	"Time to first response" > 40h
Issues that have at least one breached SLA cycle	"Time to response" = everBreached()
The order of issues based on an SLA metric	project = SIS ORDER BY "Time to resolution"

Example: creating a basic SLA

This example looks at how you might create a very basic SLA for a service desk project with a basic workflow:

Basic SLA configuration

All critical and blocker issues must be resolved within 24 hours. You provide 24/7 support for certain customers (these issues are labeled with "24H"). You provide 9 - 5 support for all other customers, but you don't track SLA metrics for them.

Time to resolution Edit

Time will be measured between the **Start** and **Stop** conditions below.

Start
Begin counting time when
Issue Created

→

Pause on
(Optional) Time is not counted during

→

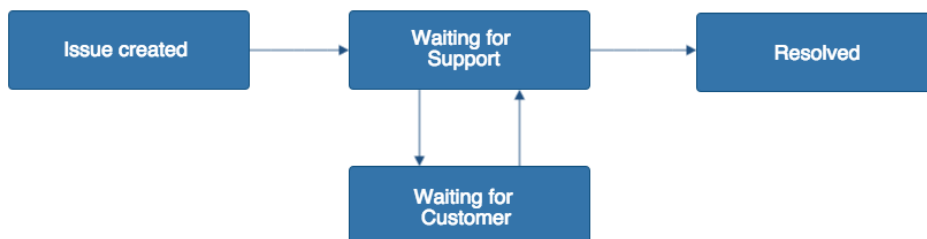
Stop
Finish counting time when
Resolution: Set

Goals

Issues will be checked against this list, top to bottom, and assigned a time target based on the first matching JQL statement.

Issues (JQL)	Goal	Calendar	Success this week
priority = Blocker or priority = Critical and labels = 24H	24 hrs	24/7 Calendar (Default)	
priority = Blocker or priority = Critical	24 hrs	9-5 working days	
All remaining issues	No target	9-5 working days	

Basic issue workflow



Example: creating an SLA that doesn't track continuous time

This example looks at how you might create a more complex SLA by pausing the time counter during the workflow:

Example SLA configuration

Support wants to complete all issues within 40 hours. Time spent waiting on the customer doesn't count against the 40 hour goal.

Time for support on all issues

Time will be measured between the **Start** and **Stop** conditions below.

Start

Begin counting time when

Issue Created

Pause on (Optional)

Time is not counted during

Status: Waiting for customer

Stop

Finish counting time when

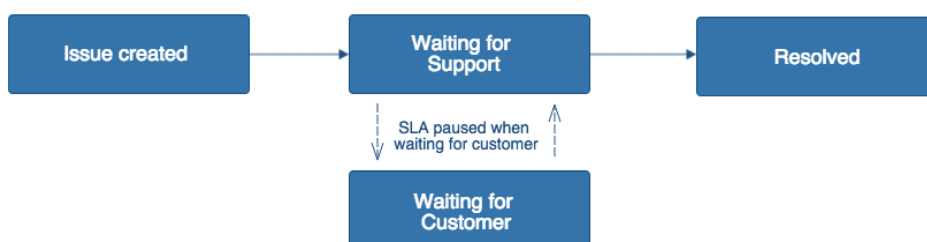
Resolution: Set

Goals

Issues will be checked against this list, top to bottom, and assigned a time target based on the first matching JQL statement.

Issues (JQL)	Goal	Calendar
All remaining issues	40h	24/7 Calendar (Default)

Example issue workflow



Example: creating an SLA with multiple cycles

This example looks at how you might create a more complex SLA by starting and stopping the time counter throughout the workflow. You might set up an SLA like this to track response times (for example, how long it takes your team to respond each time a customer updates an issue with more information). This example also illustrates how goals for different issue criteria can be tracked from a single SLA.

Example SLA configuration

Support wants to respond to Access issues within two hours: this includes responding within two hours when the issue is created, as well as each time the issue is updated with more information from the customer.

All other issues have a response time goal of 24 hours.

Time to respond

Edit

Time will be measured between the **Start** and **Stop** conditions below.

Start

Begin counting time when

→

Pause on

(Optional) Time is not counted during

→

Stop

Finish counting time when

Entered Status: Waiting for Support

Issue Created

Resolution: Set

Entered Status: Waiting for Customer

Goals

Issues will be checked against this list, top to bottom, and assigned a time target based on the first matching JQL statement.

Issues (JQL)	Goal	Success this week
issuetype = "Access"	2 hrs	
All remaining issues	24 hrs	

Example issue workflow



For further information about how SLAs with multiple start and stop conditions appear in the SLA tracker, see [Setting up SLAs](#).

Example: creating SLAs based on due dates

This example looks at how you might create a very basic SLA for a service desk project with a basic workflow:

Basic SLA configuration

All critical and blocker issues must be resolved within 24 hours. You provide 24/7 support for certain customers (these issues are labeled with "24H"). You provide 9 - 5 support for all other customers, but you don't track SLA metrics for them.

Time to resolution

Edit

Time will be measured between the **Start** and **Stop** conditions below.

Start

Begin counting time when

→

Pause on

(Optional) Time is not counted during

→

Stop

Finish counting time when

Issue Created

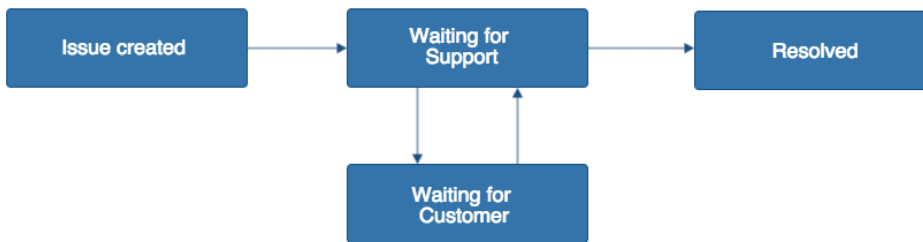
Resolution: Set

Goals

Issues will be checked against this list, top to bottom, and assigned a time target based on the first matching JQL statement.

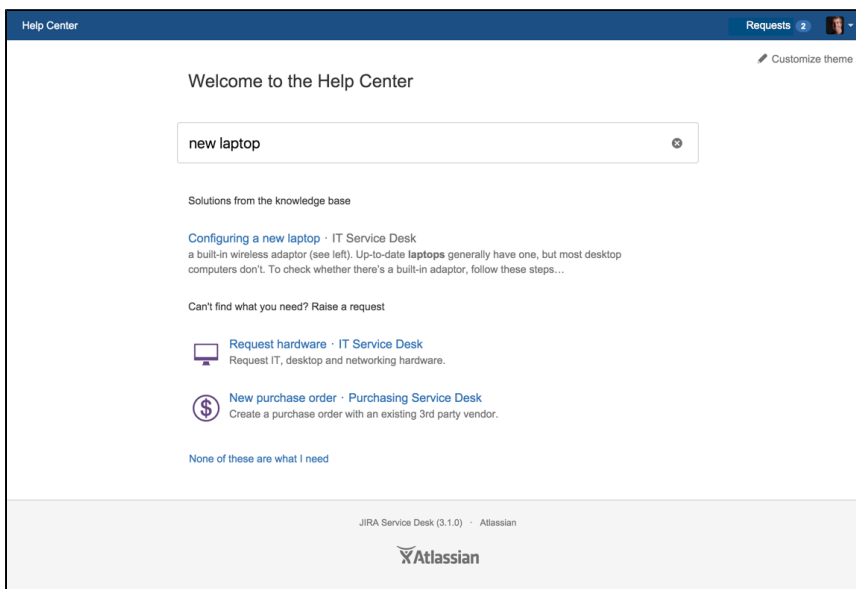
Issues (JQL)	Goal	Calendar	Success this week
priority = Blocker or priority = Critical and labels = 24H	24 hrs	24/7 Calendar (Default)	
priority = Blocker or priority = Critical	24 hrs	9-5 working days	
All remaining issues	No target	9-5 working days	

Basic issue workflow



Serving customers with a knowledge base

JIRA Service Desk can be integrated with Confluence's knowledge base capabilities, so agents can create, view, and share knowledge articles directly from the service desk issue they're working on. The knowledge articles your team creates help customers find solutions before raising requests in the customer portal or help center:



On this page:

- Integrating with Confluence
- Linking a knowledge base
- How to suggest articles for specific request types
- How agents share and create knowledge articles
- How customers find knowledge articles

Integrating with Confluence

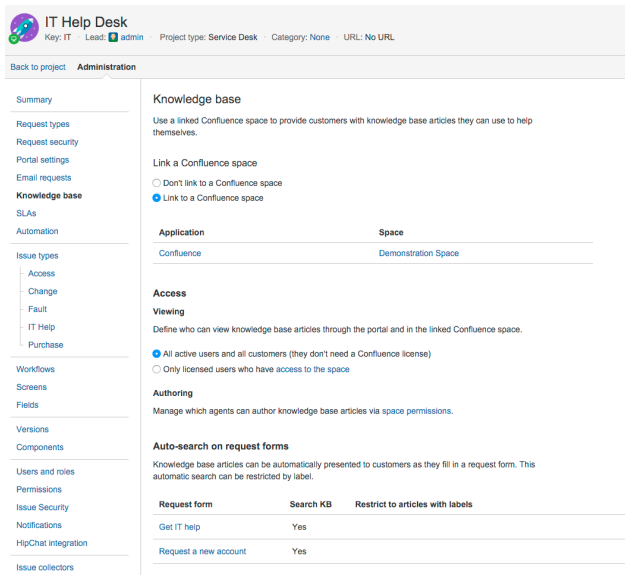
Administrators can [integrate](#) JIRA Service Desk Server with Confluence Server version 5.10 and up. They will need to link JIRA Service Desk and Confluence with an [application link](#) using OAuth.

Linking a knowledge base

Once an administrator has integrated JIRA Service Desk with Confluence, project administrators can then manage their project's Confluence knowledge base integration by going **Project settings > Knowledge base**:

Start by selecting a Confluence space to use as your linked knowledge base. Check with your Confluence administrator if you need permission to view and link a knowledge base space.

You can then choose to allow all active service desk users and customers (also known as unlicensed



Confluence users) to view your linked space. This works best if your Service Desk and Confluence instances [share a user base](#). If your Service Desk instance and Confluence instance have separate user bases, you will have to [create a Confluence user account](#) for each Service Desk customer. If you don't want the customer to consume a Confluence license, do not assign the [Confluence user to a group](#). Unlicensed Confluence users cannot:

- Like, comment on or edit Confluence content
- See the Confluence dashboard, user profiles, the people directory, or the space directly
- Search all of Confluence

If you choose to restrict viewing of the linked Confluence space to licensed users, you must grant customers a Confluence license or open your Confluence space to anonymous access. As a project administrator, you'll need to contact the administrator of your linked Confluence space to manage which agents can create and edit knowledge articles. Keep on reading to learn more about the auto-search settings you can control.

How to suggest articles for specific request types

When customers select a request type from the customer portal or help center, you can select what type of knowledge articles appear as they're filling in the request details. For example, if a customer is filling in a hardware request, you can have knowledge articles tagged with the "hardware" label appear to offer self-service suggestions. You can control how Confluence suggests articles for each request type in the **Auto-search on request forms** section. You can control this in two ways:

- **Prevent Confluence from suggesting pages** - Select **No** in the **Search KB** column for the request type. For example, you might not want the "Get access to a system" request type to suggest pages since users have to request access through the Customer Portal.
- **Limit the pages that will be suggested** - In the **Restrict to articles with label** column, enter the labels that must be applied to pages in order for them to appear in the suggested page list. For example, you might want to only include pages with the label "purchasing" to appear when customers enter a "Request new software" request.

Tip:

If you add label restrictions to a request type, these labels will also appear as the default labels for knowledge base articles [created from JIRA Service Desk](#) for issues based on that request type.

How agents share and create knowledge articles

Agents must have the [Add page permission](#) in the Confluence space to create a knowledge base article from an issue.

When agents work in a service desk project integrated with a Confluence space, they can view related knowledge base articles, instantly share articles with customers, or create new articles all from the **Related knowledge base articles** section of an issue:

AD-21 Return to queue

How do I connect to the file server?

Edit Comment Assign More Resolve this issue Respond to customer Admin Export

Details

Type: Service Request Status: WAITING FOR SUPP... (View Workflow)

Component/s: None Resolution: Unresolved

Labels: None

SLAs

8:00 Time to resolution within 8h ☐

-8:00 Time to first response within 40h ☒

Description

I am trying to access some documents on the file server and cannot seem to be able to connect to it. 😞

Attachments

Drop files to attach, or [browse](#).

Related knowledge base articles

I think that the intranet might be down. What do I... 🔒

Didn't find a suitable article? [Search knowledge base](#) or [create an article](#).

Activity

All Comments Work Log History Activity

People

Assignee: Unassigned [Assign to me](#)

Reporter: Alana Grant

Request participants: None

Votes: 0 [Vote for this issue](#)

Watchers: 1 [Start watching this issue](#)

Service Desk request

Request type: Get IT help

Customer status: Waiting for support

Channel: Unknown

[View customer request](#)

How to share an article with a customer

When agents see a related, unrestricted article that will help resolve a customer's request, they can simply select **Share as comment** directly from the issue or from the article dialog to create a new comment with the article name and link. Restricted articles can be viewed by agents within the service desk issue; however, they cannot be shared in customer comments.

How to create a new knowledge article

If the needed knowledge article does not appear in the knowledge base article panel or in the knowledge base search, agents can create a new article directly from the issue they're working on. When creating a new article, agents can choose to either the How-To or Troubleshooting Confluence page template.

Related knowledge base articles

I think that the intranet might be down. What do I... 🔒

Didn't find a suitable article? [Search knowledge base](#) or [create an article](#).

Activity

All Comments Work Log History Activity

How do I connect to the file server?

How-To

Create Cancel

These templates give agents content suggestions, and can help you expand your knowledge base with clearly organized topics. The issue title and description will be automatically added to the new Confluence page as its title and body text. If you've set up [label restrictions](#) for the request type the issue was based on, those labels will be automatically suggested for the article. Note that images and attachments will not be automatically copied from the service desk issue to the knowledge base article.

How customers find knowledge articles

Customers will be able to search for knowledge articles connected to service desk projects they have access to. If you have linked a Confluence space to an [open service desk](#) (with public signup enabled), then all customers will be able to view the knowledge articles in that space. Customers can view knowledge articles in the customer portal, the global help center, and in Confluence if they're provided with a direct link.

You can also help customers fill out a request form by suggesting knowledge articles [related](#) to that type of request.

Using the help center

You can direct customers to your help center, so they don't have to remember whether they need to submit a request for a new laptop in the IT Service Desk or Purchasing Service Desk. Simply searching for "new laptop" in the help center will display the correct request type automatically.

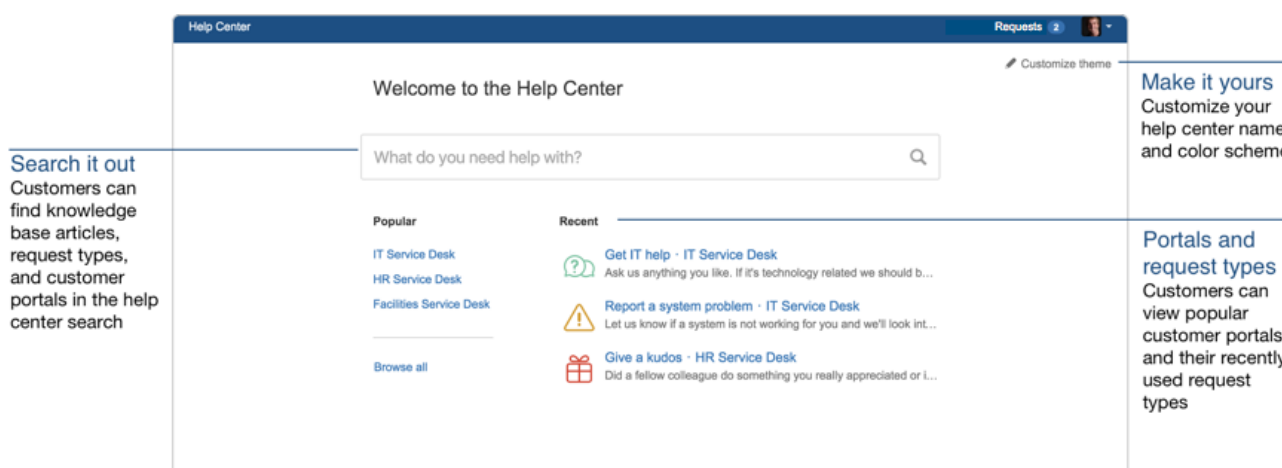
Customers can use the help center to:

- View popular service desks
- Browse an individual service desk
- Search for request types and knowledge base articles
- See requests they have raised across all service desks

On this page:

- [Branding the help center](#)
- [Sharing the help center with customers](#)
- [Searching across multiple customer portals](#)
- [How smart search works](#)

To help your customers get the most out of your help center, we recommend branding your help center and [integrating your service desk projects with a knowledge base](#). Here's a quick look at the help center layout:



Customers will only be able to see popular service desks and search across service desks they already have access to. The list of popular service desks is generated automatically based on the number of requests raised and cannot be set manually. Recent request types displayed are unique to each customer. Note that customers who have not yet raised requests will not see any recent request types in the help center. Learn more about managing customer access to service desk projects [here](#).

Branding the help center

JIRA administrators can brand the global center with your company logo and color scheme. If you are logged in as an administrator, go to



> **Applications > JIRA Service Desk Configuration** to customize your help center with the help of a live preview. Your changes will be applied to the help center and to the header of all customer portals. For more information about managing a project-specific customer portal, check out [Configuring the customer portal](#).

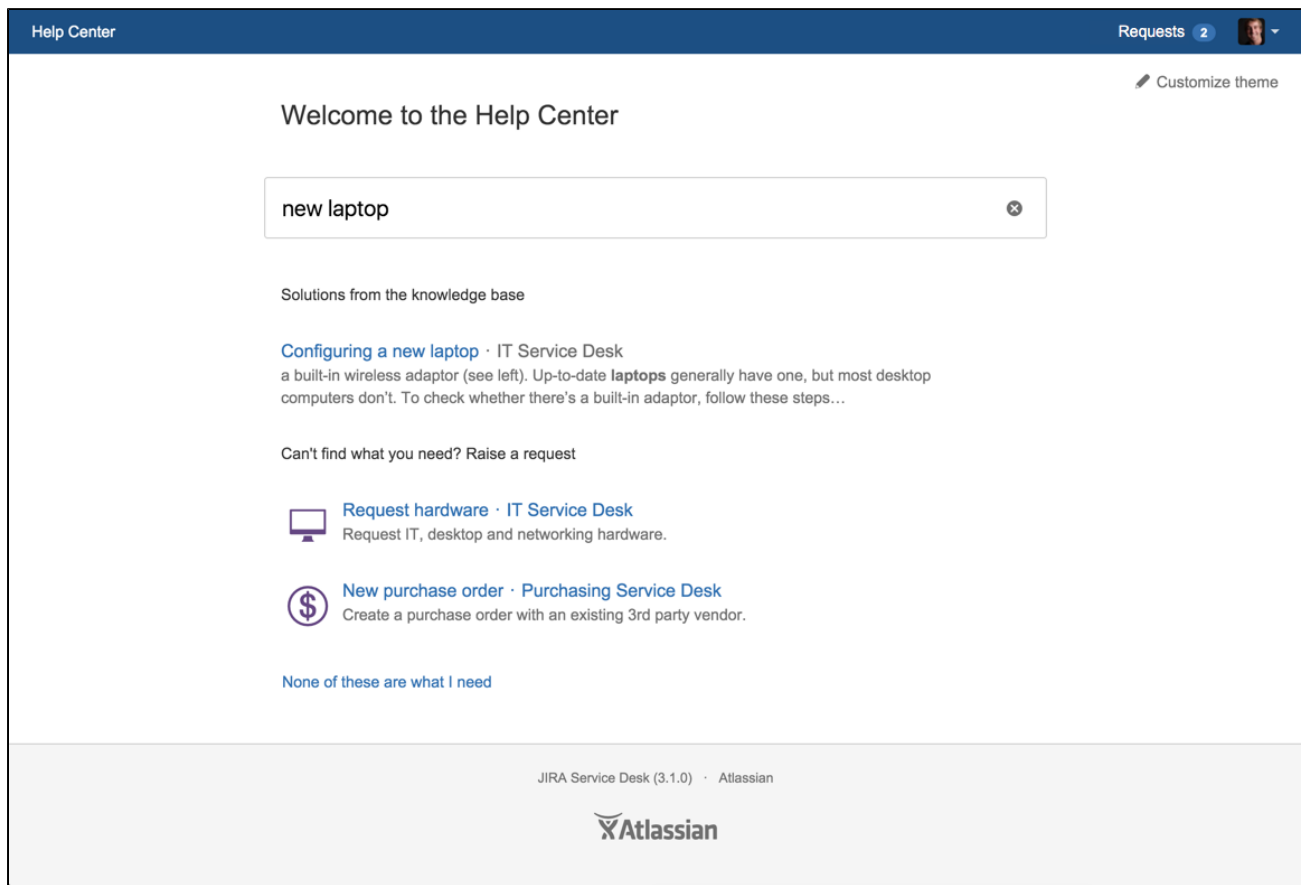
Sharing the help center with customers

You can provide your customers with the following help center URL, so they don't have to remember the URL for each customer portal they have access to:

`http://<computer_name_or_IP_address>:<HTTP_port_number>/jira/servicedesk/customer/portal/<portal_id>`

Searching across multiple customer portals

Customers can search for request types and knowledge articles in a specific customer portal or use the global help center search to find request types and knowledge articles in portals they have access to. Let's use the same new laptop example. When customers search for "new laptop" in the help center, they'll find related knowledge articles and request types, as shown here:



Customers can read articles directly in the help center and provide feedback to your team by marking articles as helpful or not helpful. If customers still need to contact your team after looking through related knowledge articles, they can choose one of the suggested request types or browse all customer portals they have access to.

How smart search works

The improved smart search algorithm learns from past searches and request types raised, so if customers have previously raised hardware requests for a laptop and monitor, they can search for "laptop" or "monitor" in the future to find the same hardware request type. Your team can also help improve search results by updating the request type field when a customer has, for example, searched for "new laptop" and raised a purchase order request instead of the needed hardware request.

The help center smart search has been built to be language-agnostic and can therefore learn from search words or phrases entered in any language. As customers enter more searches and raise more requests, the search algorithm gets smarter regardless of the language used.

Collecting customer satisfaction (CSAT) feedback

Measuring customer satisfaction can help you better understand your customers and improve service levels.

JIRA Service Desk provides a simple, built-in mechanism to collect customer feedback. Key features include:

- Simple customer workflow to provide feedback on resolved issues
- Customer satisfaction scores are visible within resolved issues and on agent queues for resolved issues
- Single-click to view customer satisfaction report for a service desk

- project
- Easily create and view custom reports and trend graphs based on satisfaction scores.

Customer feedback data can be used to identify strengths and weaknesses in the service quality, engage and motivate the team to improve satisfaction scores, and provide mentoring and training where required.

On this page:

- [Enabling the customer satisfaction feature](#)
- [Viewing and reporting on customer feedback](#)

Enabling the customer satisfaction feature

This feature is enabled by default for new service desk projects; however, it must be enabled for each existing service desk project. To enable customer satisfaction settings for an existing project:

1. Log in as a service desk project administrator.
2. Select the service desk project you wish to configure.
3. Select **Project settings > Satisfaction settings**.
4. Enable **Collect customer satisfaction feedback**.
5. Optionally, edit the **Question** phrase to suit your service desk environment. This phrase appears in the resolved issue notification message that customers see.

When you enable satisfaction settings for a service desk project, resolved issue notifications will contain a satisfaction rating scale. Customers can click the rating scale to indicate their level of satisfaction. A confirmation page is displayed on the customer portal, where they can change the rating, and optionally provide any additional comments that they would like to convey to the team.

Hi Emma,

Your request has been resolved.


– José

How was our service for this request?

☆ ☆ ☆ ☆ ☆

See request details and updates for [IT-458](#) - "Can't print to level 8 printer"

Help Center sent you this message, powered by JIRA Service Desk


 ★ ★ ★ ★ ★
Awesome! We got it.
 If you have a moment, tell us how it went so we can keep it up!




Thanks for helping me so quickly!

Add a comment

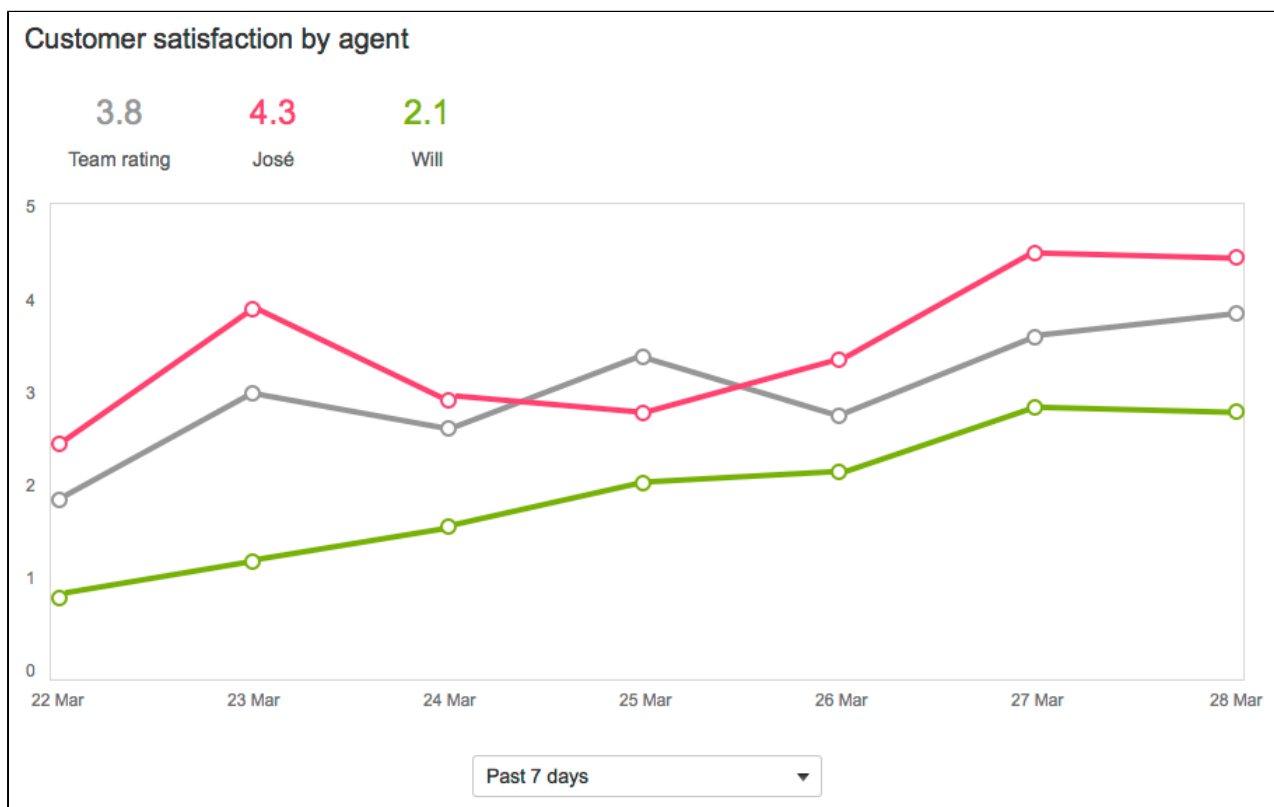
Viewing and reporting on customer feedback

Customer satisfaction scores and comments are displayed in the issue view for resolved issues. Agents can also view the satisfaction scores on their own recently resolved queues.

Service desk project administrators and agents can view the default Satisfaction report, which displays the average customer satisfaction scores for the team.

Customer satisfaction				
<div>2.6</div> <div>Average rating</div>				
Rating	Comment	Key	Agent	Received
★★★★★	Thanks for coming down to my desk to help me print in A3	IT-1384	 Will Turner	Yesterday
★★★★★		IT-997	 José Vela	2 days ago
★☆☆☆☆	I had to wait 3 days for a solution. I replied on Wednesday but nobody got back to me until this morning.	IT-1216	 Mia Kennedy	3 days ago
1-3 of 3				
<div>Past 7 days</div>				

A service desk project administrator can also create and view [custom reports](#) analyzing satisfaction trends. Agents can also view any custom satisfaction reports created for their service desk projects.



Useful examples of custom reports include:

- A trend graph of the average satisfaction rating for a specific period to view changes in service levels.
- Satisfaction scores based on the type of service request. This would identify issues for which the team could provide knowledge articles.
- Satisfaction scores for an individual agent compared to team scores to help identify agents who would benefit from further training.

JIRA Service Desk best practices

Check out the following best practice articles:



- Best practices for designing the customer portal
- Best practices for IT teams using JIRA Service Desk

Best practices for designing the customer portal

Every service desk project comes with a preconfigured customer portal that your customers use to interact with your service team. Here are some best practices on how to design an easy-to-use customer portal that will help both your team and your customers work more efficiently.

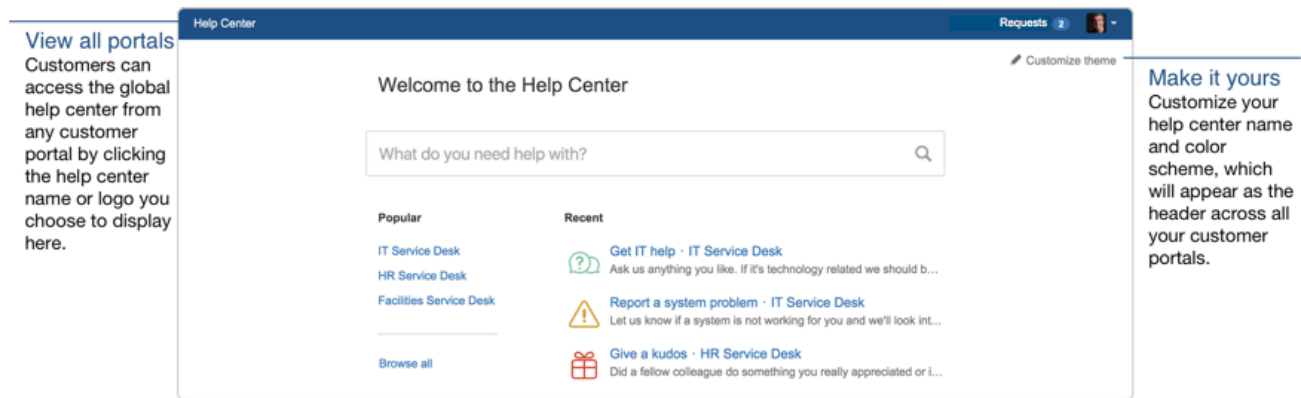
On this page:

- Brand your help center
- Brand your customer portals
- Help customers find the request types they need
- Group related request types
- Set up a knowledge base

Brand your help center

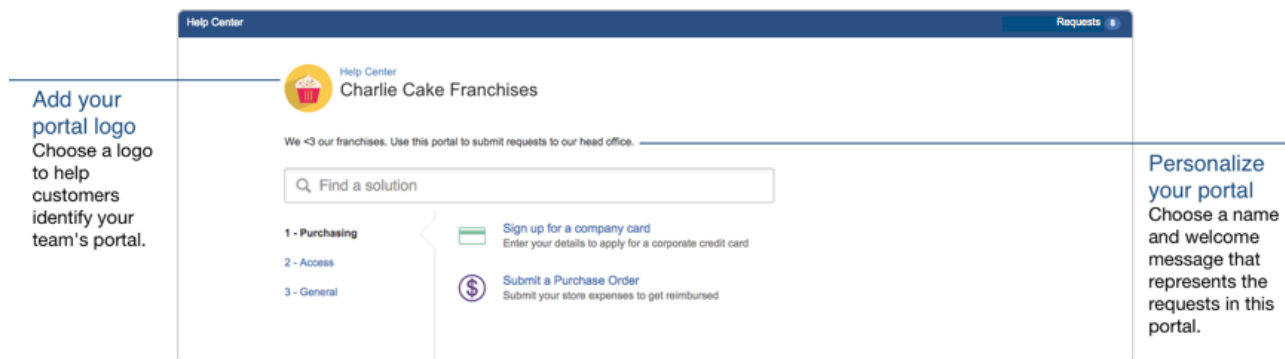
Customers can use the global help center to search for request types and knowledge base articles across all customer portals they have access to. Brand your help center by:

- Uploading your company logo and your service desk can automatically generate a matching theme for your customer portals and global Help Center header.
- Naming your customer portals and global Help Center, so your customers can easily identify your team's service desk.



Brand your customer portals

For each project's customer portal, you can customize the name, welcome message, and logo to let customers know which portal to use to contact a specific team in your organization. Note that the help center name and header appear across all your project's customer portals.



Check out [Configuring the customer portal](#) to learn more.

Help customers find the request types they need

- Name request types in language that's familiar to your customers, and use keywords they'll recognize. For example, name a request 'Access to a system' instead of 'VPN access'.
- Use different icons for different request types, so customers can easily identify request types in the customer portal.
- Add contextual help (e.g. specify photo dimensions and format for the attachment field) with the **Field help** field.
- Use examples in your request type descriptions (e.g. 'If you need a software license such as Microsoft Office, raise a request here').
- Link to existing information that might be helpful for customers in the request type description. For example, if you have already have a list of available Microsoft Office license numbers on your Intranet, simply add a link to the page in the request type description and instruct customers to claim a license from that page without needing to open a request.

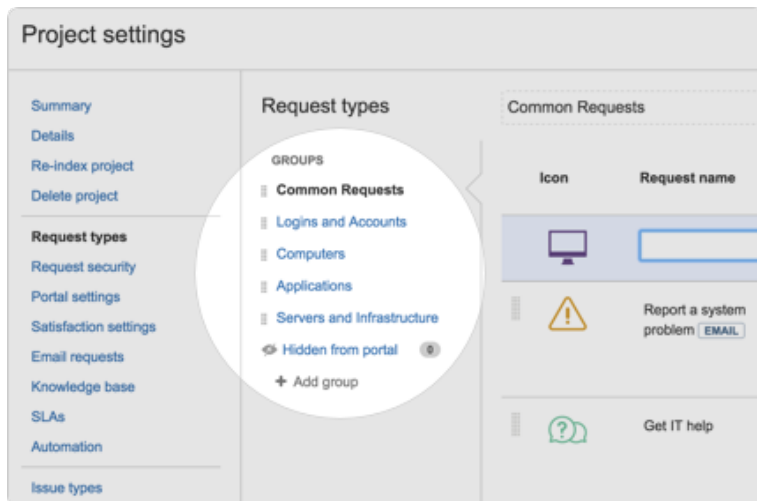
See [Setting up request types](#) for more information on naming request types.

Group related request types

If you have a large number of request types, say more than 7, we recommend grouping some of them together to help customers find what they need. Grouped request types appear as tabs in your customer portal. To add groups, select **+Add group** from the sidebar. While viewing a group, select **Add existing request type** to add your request types to it.

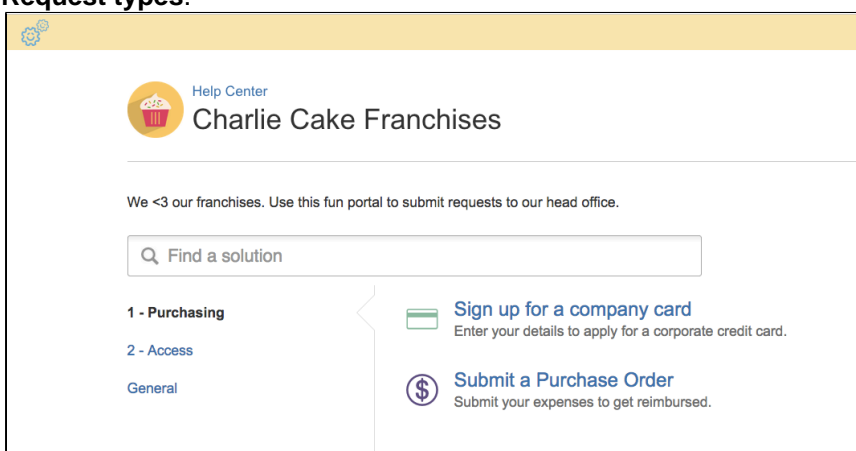
If you don't want your customers to use a certain request type, you can remove the request type from all groups, or move it into the hidden from portal group.

Agents can use



hidden request types to organize work internally. Note that removing a request type won't affect requests that have already been created.

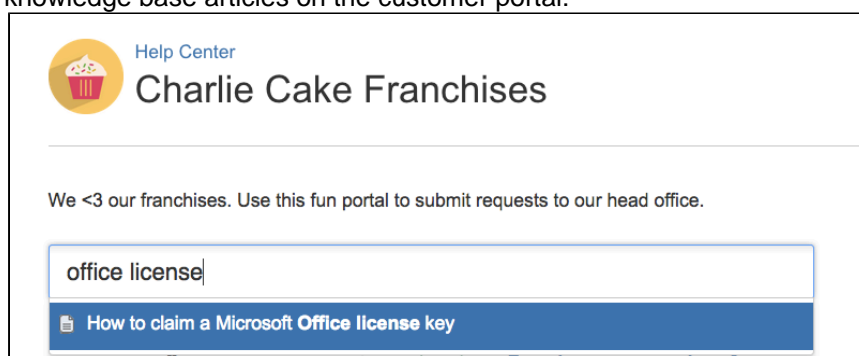
Groups appear as tabs in your customer portal. You can control the order in which groups appear by dragging and dropping them in **Project settings > Request types**.



See [Setting up request types](#) for more information on managing request types and groups.

Set up a knowledge base

- After using your service desk for a while, your team will probably have accumulated a large amount of information that can be provided to your customers so that they can solve some problems before even opening requests. At this point, you can consider integrating Confluence's knowledge base capabilities with JIRA Service Desk.
- Connect your service desk project to a Confluence space so customers can search for relevant knowledge base articles on the customer portal:



For information about how to achieve this, see [Serving customers with a knowledge base](#)

Best practices for IT teams using JIRA Service Desk

Your team can manage service requests, incidents, problems and change in JIRA Service Desk easily. By the end of this guide, you will have reviewed

the default setup provided by the IT Service Desk template for these processes, and be ready to start configuring the forms and workflows for your own requirements.

JIRA Service Desk has a template for IT teams and this template comes with preconfigured request types, issue types, queues, workflows, and reports to help you and your team get started with managing the processes immediately.

The related pages (listed on the right) provide practical ITIL insights and process examples to assist with your JIRA Service Desk implementation.

This page

- [Overview](#)
- [Setting up a project](#)
- [Have a tour of the setup in the project](#)
- [Creating service requests, incidents, problem and change records](#)
- [Tracking changes and keeping everyone in the loop](#)
- [The power of Confluence knowledge base](#)
- [Success! Customize the setup](#)

Related pages

- [Service request fulfillment](#)
- [Change management](#)
- [Problem management](#)
- [Incident management](#)

Overview

Many IT organizations have chosen to adopt ITIL as their standard for running their IT organization. ITIL (Information Technology Infrastructure Library) is the gold standard for IT organizations worldwide because it provides the framework for IT service management (ITSM) that focuses on aligning IT services with the needs of the business. The ultimate goal of ITIL is to improve how IT delivers and supports business services. ITIL is not just technology management or process management, it also focuses on improving the capabilities of people, processes, and technology, while delivering value to an organization.

Adopting ITIL is a journey that takes time and commitment.

JIRA Service Desk offers IT teams a streamlined approach for adopting ITIL in a way that best fits the needs of their IT organization. You can easily adapt JIRA Service Desk to meet your ITIL requirements without

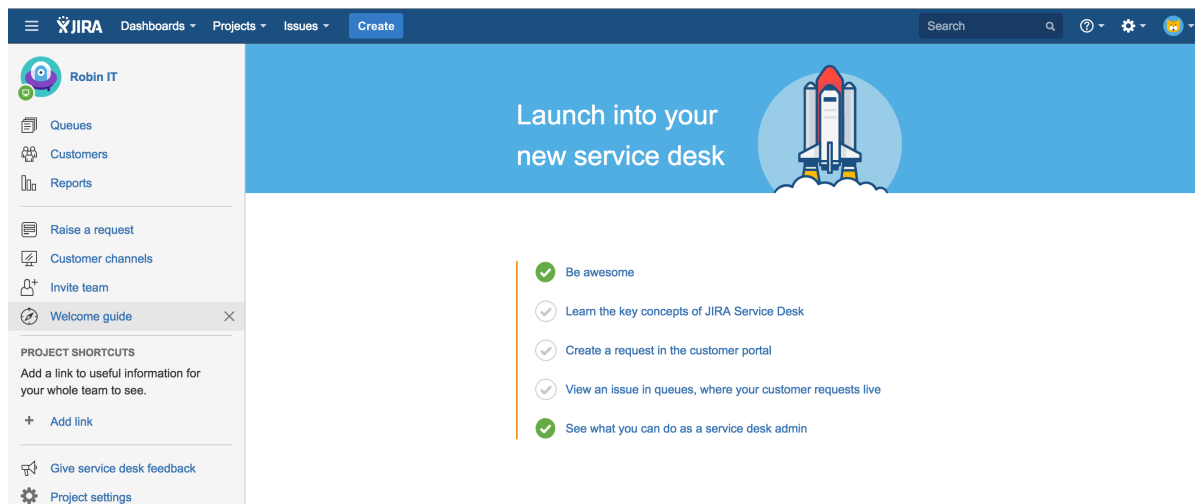
compromising the ease of use your IT team expects. The IT project template pre-packages the setup for service requests, incident management, problem management and change management, but we suggest that you treat the implementation with an agile flavor:

1. Start with defining the most common and urgent types of work that need to be tracked in JIRA Service Desk and implement those first.
2. Roll them out, provide value to the business and gather feedback.
3. After that, continue building upon the existing configuration and refine the setup.

Setting up a project

In this step, you'll create a new IT Service Desk project with the IT template.

1. Sign into JIRA Service Desk as an administrator.
2. Select **Projects** > **Create Project**.
3. Choose the **IT Service Desk** template and select **Next**.
4. Name your project, e.g. "IT Sample". You will see the project key field automatically filled in. This project key is the first part of the unique ID for each of the records in your project. The following example uses the key of 'ITSAMPLE'.
5. Select **Submit**. You will land in a service desk similar to the following:

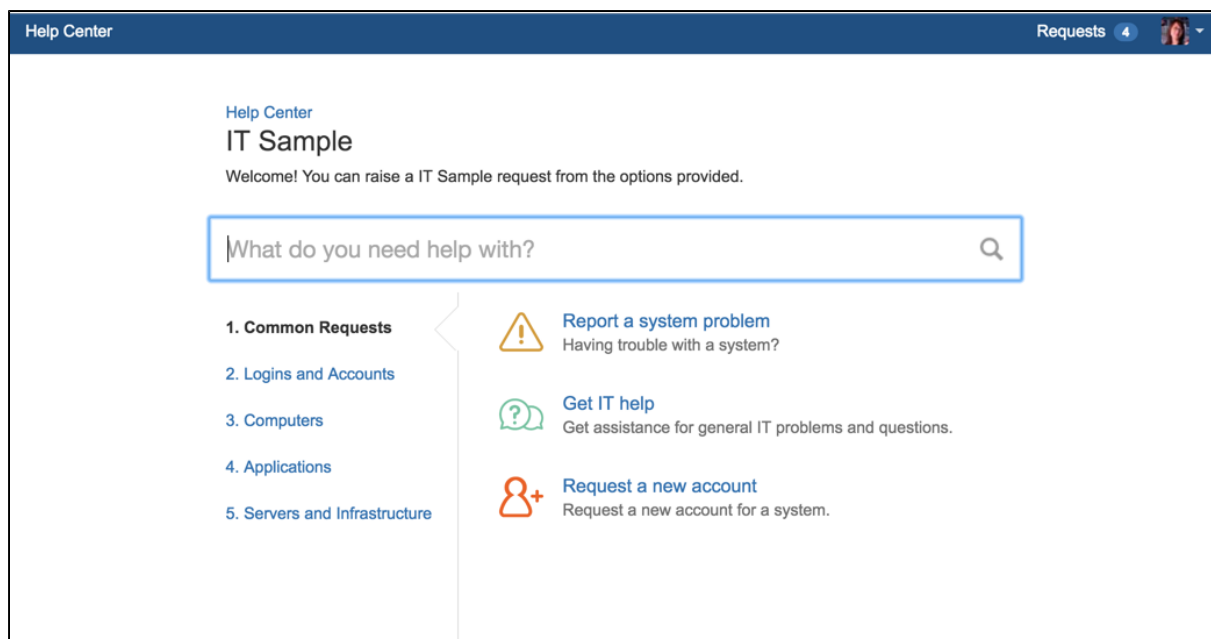


6. Optional: If you are new to JIRA Service Desk, have a look at [Getting started with JIRA Service Desk](#).

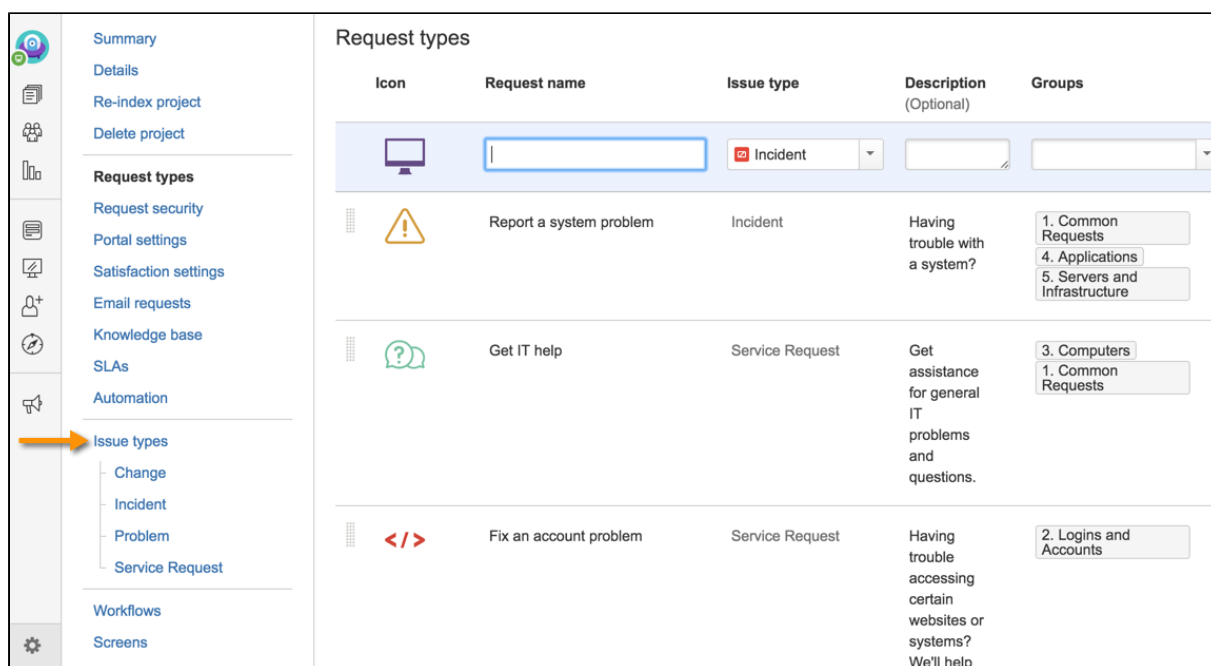
Have a tour of the setup in the project

Now let's get familiar with the default setup in the project for service requests, incidents, problem and change.

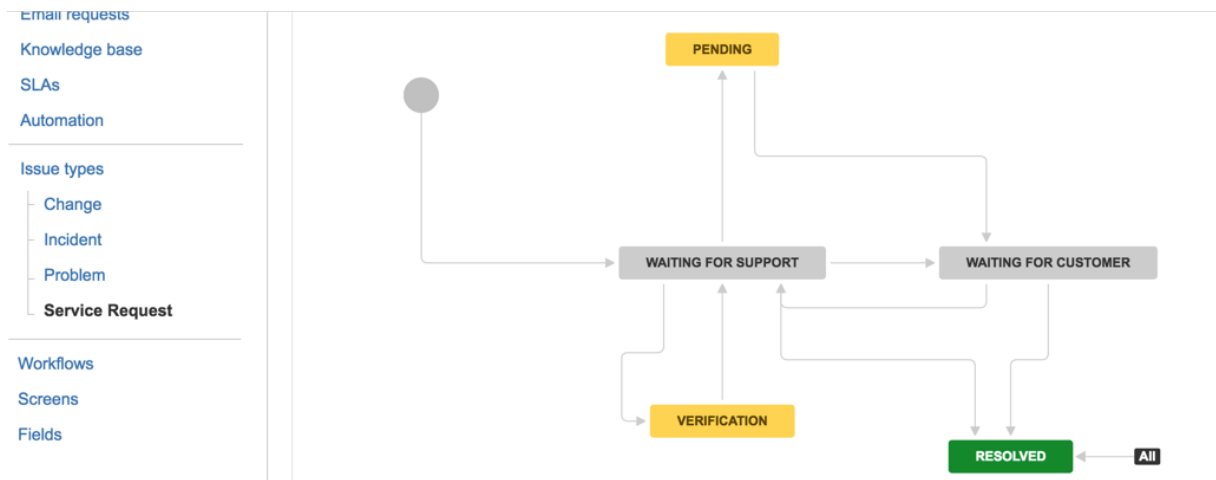
1. Have a look at the customer portal. It shows the request types defined in the template. Request types are a lightweight way to set up a service catalog in JIRA Service Desk.



2. In the sidebar, go to the **Project settings** section at the bottom. You will see the request types that you saw on the customer portal in the last step. Each request type is associated with an issue type, and JIRA Service Desk defines four different issue types for the four processes. Find the **Issue types** section in the settings and you'll see **Change**, **Incident**, **Problem**, and **Service Request**. For more information, see [Setting up request types](#).



3. Review the default workflows for each process. For example, the following screenshot shows the workflow for service requests. You will also be able to see the fields used by each process.



Creating service requests, incidents, problem and change records

There are multiple channels available for the creation of those records:

- **Opening service requests:** Your customers can open service requests easily through the customer portal and via email. For ITIL-based recommendations, see [Service request fulfilment](#).

Opening a service request in the customer portal

- **Reporting incidents:** Incidents can be reported from different sources, via the customer portal, email, or system alerts with the [REST API](#). Again, in JIRA terminology, every incident record is tracked as an *issue*. When an incident is reported, it's allocated a distinct identifier called an *issue key*.

Help Center

Help Center / IT Sample

Report a system problem

Summarize the problem

Intranet seems to be really slow

Description (optional)

Attachment (optional)

Drag and drop files, paste screenshots, or [browse](#)

Select a system (optional)

x Intranet

How urgent is this? (optional)

Medium

[Create](#) [Cancel](#)

Reporting an incident in the customer portal

- **Requesting changes:** Similarly, requests for change can also be created via the customer portal, email, or programmatically with the [REST API](#).

Help Center

Help Center / IT Sample

Upgrade or change a server

Summary

Which server and why?

Requesting change in the customer portal

- **Creating problem records:** Typically, problems are not created by customers and that's why the default setup doesn't include any problem request types on the customer portal. The agents can generate a problem from within JIRA by clicking **Create** and selecting the **Problem** issue type.

Tracking changes and keeping everyone in the loop

JIRA Service Desk automatically tracks all the changes to an issue with date and time stamps, and these details are visible in the **Activity** and **History** tabs of the issue. Email notifications are sent to the people involved in an incident and an administrator can configure these notifications for their processes. For information about this, check out [Managing service desk notifications](#).

The power of Confluence knowledge base

Customers like to self-service these days, and your agents need to be able to find known errors, work-around s, temporary fixes and routine fixes right quickly and efficiently. You can achieve both of these goals by integrating JIRA Service Desk with Confluence, and using Confluence to serve a knowledge base for JIRA Service Desk. For information about this, see [Serving customers with a knowledge base](#).

Success! Customize the setup

We know that each IT or service team is unique, and the default workflows, fields, request types and queues for the four processes are only provided as a starting point for you to build your own. Refer to the setup information for instructions: [Administering service desk projects](#).

The following pages provide information about each individual process, and a sample workflow diagram based on the ITIL framework. Feel free to refer to these samples as you set up your own:

- [Service request fulfillment](#)
- [Incident management](#)
- [Problem management](#)
- [Change management](#)

Service request fulfillment

What is a service request?

IT receives a wide variety of requests from customers. These requests may include simple requests for support, a new mobile device, software installation, or even a request to move office desktop equipment. ITIL classifies these types of requests as a 'request for service'. The size, frequency and low risk nature of these types of requests means that they are more appropriately handled by a separate process, rather than with incident or change management processes. ITIL identifies the process to

handle service requests as *request fulfillment*. This process is intended to streamline the fulfillment of service requests while delivering the highest level of service support quality to customers.

Many service requests will be recurring, so to achieve the greatest efficiency a repeatable process and procedure should be defined. While some variations with the fulfillment of service requests may exist, it's important to adopt a standard set of processes for request fulfillment. The ownership of service requests resides with the service desk, which monitors, communicates with the customer, escalates, and often times fulfills the service request.

This page

- [What is a service request?](#)
- [What is request fulfillment?](#)
 - [Benefits of request fulfillment](#)
- [Service request fulfillment process](#)
 - [Fields](#)

Related pages

- [Incident management](#)
- [Change management](#)
- [Problem management](#)

What is request fulfillment?

Request fulfillment is the process for fulfilling a customers request for service. Some of the key objectives of the request fulfillment include:

- Separate request fulfillment from the Incident and Change processes.
- Request fulfillment needs to be driven by standard services that are defined in the service catalog.
- Provide a single point of contact for requesting standard services from an online service catalog.
- Standard requestable services should include pre-defined processes that streamline the fulfillment.
- Clearly communicate available services, the procedure for requesting them, and the expected fulfillment timeframe.

The process needed to fulfill a request will vary depending upon what is requested, but can usually be broken into a set of activities that have to be performed. In an organization where large numbers of service requests have to be handled, and where the actions to be taken to fulfill those requests are varied, it may be appropriate to handle service requests as a completely separate work stream – and to record and manage them as a separate record type. Since the service desk team is the first point of contact for customers it recommend they act as the first line of contact for handling service request. Many service requests can and should be handled by the service desk so long as they have sufficient resource, time, tools and skills.

Benefits of request fulfillment

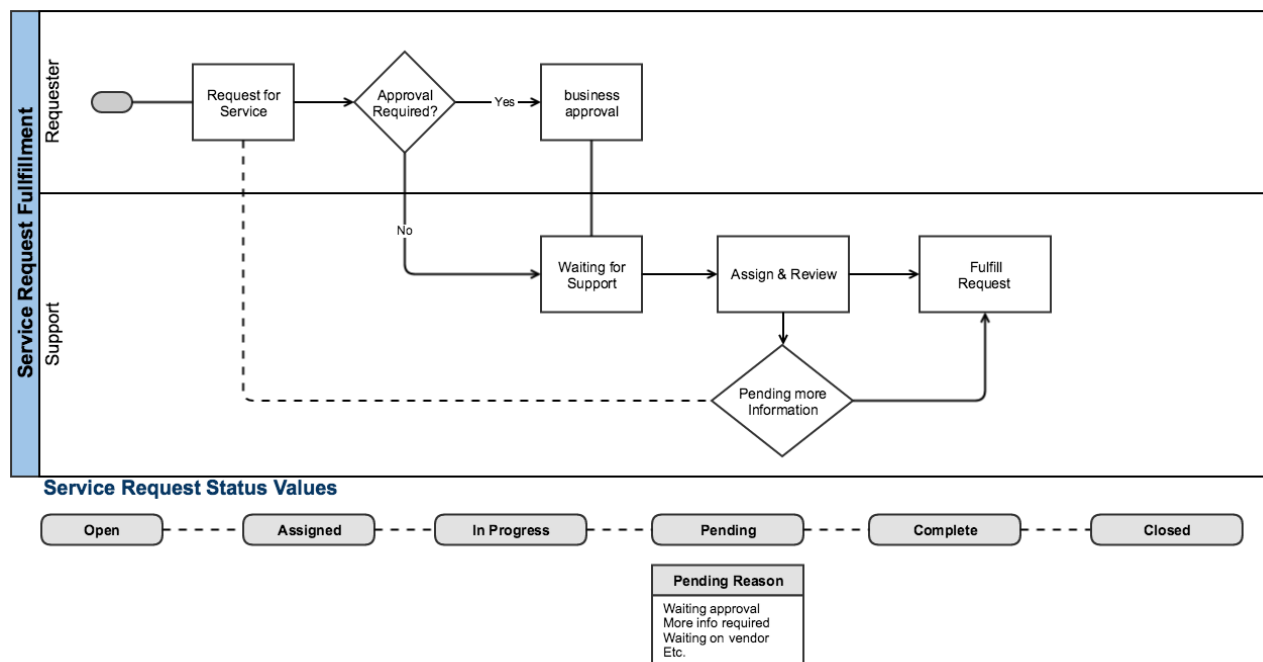
Request fulfillment provides customers with quick and effective access to standard services. Request fulfillment reduces the workload involved in requesting and receiving access to the services that IT provides. Providing this type of access plays an important role in reducing the cost of providing these services. Centralizing the process also increases the level of control over these services. This centralized approach

also provides a great opportunity to share knowledge through a searchable knowledge base that provides customers with the information they need. This helps deflect requests and deliver immediate value to the customer.

Request fulfilment also shows the value that support organizations provide to the business by demonstrating the actual cost of the services they offer and providing insight into the resources that are needed to provide certain kinds of services.

Service request fulfilment process

While you will find some variations in the way a service request is captured and fulfilled, it is important to focus on ways to drive standardization to improve the overall service quality and efficiency. The following process represents a sample for request fulfilment based on ITIL recommendations. You can use this simple request fulfilment process as a starting point for adapting existing ITIL processes or defining new ones.



Important request fulfillment activities to consider when defining service requests for JIRA Service Desk

- Begin with the most commonly requested items. Choose ones that are simple and easily fulfilled. This delivers immediate value to customers and allows the IT service desk team to learn as they build out future phases of the request catalog.
- Document all service request offering's requirements (question data, approval process, fulfillment procedures, fulfillment team, process owner, SLAs, reporting, etc.) before you add them to the request catalog. This will allow the IT team to best manage the request offering over time. This step is very important for more complex request offerings that will evolve over time.
- Capture the data needed to start the request process, but don't overload the customer with too many questions.
- Standardize the approval process where possible. For example, all requests for new monitors are considered pre-approved, and all software requests need to be approved by the customers' manager.
- Review the request fulfillment process and procedures to identify which support teams are responsible for completing the request, and if any special requirements exist.
- Identify what knowledge information should be available in the Knowledge Base when a request offering is released. The overall goal of self-service is to give your customers what they want faster and to deflect requests where possible, so if you can answer a question in a common FAQ, include this knowledge as a part of the plan when creating the service request offering.
- Review Service Level Agreements to ensure you have the proper measurements and notifications in place so that requests are fulfilled in a timely manner.
- Identify what reporting is needed to properly manage the life cycle of a service request and request offering in the catalog in the long term.

Fields

We recommend the use of the following fields for your request fulfilment process:

- **Description:** Use this field to capture the basic information about the incident.
- **Status:** Indicates the current state of the service request.
- **Pending Reason:** Specifies the reason for why service requests are moved into pending and what they are pending on.
 - **Example values:** Waiting on vendor, More info required, Awaiting approval,
- **Priority:** This is determined by the urgency and impact of the request. Your team can define the value matching according to your own processes.
 - **Example values:** Critical, High, Medium, Low
- **Urgency:** A measure how quickly a resolution of the request is required
 - **Example values:** Critical, High, Medium, Low
- **Impact:** A measure of the extent of the Incident and of the potential damage caused by the request before it can be resolved.
 - **Example values:** Extensive / Widespread, Significant / Large, Moderate / Limited, Minor / Localized
- **Operational categorization:** Classifies a request for the purpose of assignment and reporting from the operational perspective.
 - **Example values:** Configuration > Printer
- **Product categorization:** Classifies a request for the purpose of assignment and reporting from the product perspective.
 - **Example:** Hardware > Printer
- **Component:** Indicates the service associated with the request.
- **Resolution:** Classifies the resolution.

Change management

What is change management?

Every IT organization faces the challenge of managing a constantly changing IT infrastructure. IT needs to roll out new technologies while managing existing ones, along with a never ending churn of updates and upgrades. Demanding regulations also require IT organizations to provide a detailed audit history of these IT system changes. Close control of these changes helps IT keep track of both historical and current changes. To be successful with change management, IT needs a repeatable process that captures the appropriate data, enforces the proper change rules, while keeping the whole process streamlined and easy for the IT team to use. ITIL provides a set of best practices for change management that makes it easier for IT teams to prioritize and manage changes efficiently, while minimizing the impact on the business. Change management is a process designed to understand and minimize risks while making IT changes to critical systems and services. The business has two main expectations of the services IT provides:

- Services should be stable, reliable, and predictable.
- Services need to be adaptable to rapidly evolving business requirements.

Change management applies a formal process to accomplish change and is sometimes thought of as making change more difficult by creating more work for the IT team. But a properly implemented change management process enables a streamlined approach to capturing change requirements while enabling the proper governance needed for success.

This page

- [What is change management?](#)
 - [Benefits of change management](#)

- Change management process
 - Change management roles
 - Change request types
 - Change Management process summary

- Setup for change management in JIRA Service Desk
 - Configure the workflow and fields with the Change Management workflow add-on
 - *To display these fields for your
 - Change management fields
 - Change scheduling

Related pages

- Service request fulfillment
- Problem management
- Incident management

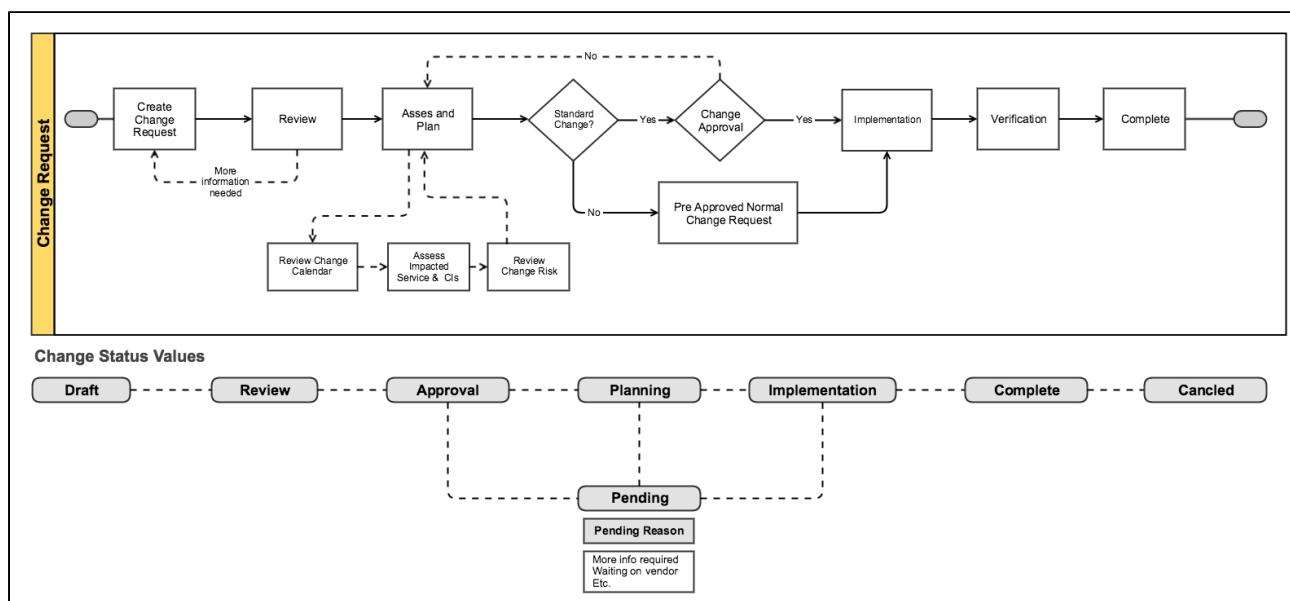
Benefits of change management

Adopting a standardized ITIL approach to change management enables an IT organization to deliver significant improvements. An effective change management process provides the following benefits.

- Provide a standardized approach for prioritizing and responding to business requests for change
- Reduce failed changes and service disruption, defects and rework
- Deliver change promptly to meet business timescales
- Provide better estimations of the quality, time and cost of change
- Comply with all governance, legal, and regulatory requirements for tracking changes to IT systems
- Improve the overall productivity of IT teams by minimizing disruptions due to high levels of unplanned or emergency change

Change management process

This process represents a streamlined approach to a request for change based on ITIL recommendations. It address the two most common types of change requests: standard and normal. This example provides a starting point from which you can adapt your existing processes and support procedures.



Changes management roles

How your IT team is structured to manage change requests will vary by IT organization, but there are a couple of key roles to consider when defining your support process and procedures.

The **change requestor** is an IT team member that works with IT systems who provides support services for customers. When they recognize a need for a change to an IT system, service or application, they will raise a formal request for change. This request should be fully documented and clearly indicate the service and system impacted by the change.

The **change manager** is the one responsible for managing the change procedures. As part of their responsibilities, they review and prioritize all change requests, and evaluate their change risk level. In some

IT organizations, they may provide the initial level of approval before the change is allowed to move into the detailed planning stage. The change manager is responsible for scheduling the change implementation date based on the use of a centralized change calendar. They also review all changes once they are complete to monitor the overall quality of the change process.

The **change advisory board (CAB)** is responsible for reviewing and authorizing change requests when the change manager has determined there is a high risk associated with it. The CAB takes into account the impact that a change request may have on the business and all affected organizations. When high-risk changes occur it's the responsibility of the change manager to communicate the appropriate information about the change to the CAB. In an effort to keep the change process streamlined and efficient, the use of a virtual CAB may be an effective means of managing this part of the process. A virtual CAB leverages the use of technology to communicate the high-risk change and indicate their approval or disapproval with ease in the change record. In the case of some high-risk production changes a formal meeting may be required to determine how to proceed.

The **change implementation team** consists of the specialists across your IT organization who are responsible for actually making changes. You will likely be part of this team and employees directly under you may also be assigned to implement changes. The change implementor is responsible for completing the change request and documenting all work completed in the request.

Changes request types

There are three different types of change requests that are typically managed in different ways:

Standard change

Standard changes are pre-approved changes that are considered relatively low risk, are performed frequently, and follow a documented (and change management approved) process. Think standard as in, 'done according to the approved, standard process'. Not standard, as in run-of-the-mill. The process for a proposed standard change is presented to change management to review/approve. The proposed standard change describes how the change and associated risks will be managed. Once change management has approved the standard change, it can be carried out in production as needed. It's worth noting that standard changes, even after approval, are still under the jurisdiction of change management. If standard changes start causing incidents, change management can bring the standard change back for review and request changes as needed.

The primary purpose of standard changes is to provide a way of streamlining the process to enable rapid implementation of frequent changes while managing the risk. A standard change must have a documented process that's been reviewed and approved by change management.

Normal change

Normal changes are the run of the mill not 'standard' and non-emergency changes that require change management review. They are raised as Request for Change (RFC), reviewed by CAB (Change Approval Board), and approved or rejected by the Change Manager. Normal changes are often non-trivial changes to services, processes, and infrastructure.

Emergency change

Emergency changes arise when an unexpected error or threat occurs, such as when a flaw in the infrastructure related to services needs to be addressed immediately. A security threat is another example of an emergency situation that requires changes to be made immediately.

Change Management process summary

The ITIL change management process includes many vital steps to ensuring a change request is successfully completed. Here's a summary of the vital steps included in a change management process:

1. **Request for Change** - the first question to consider in your IT organization is who should be allowed to request a change and what is the proper procedure for creating one. You want a simple and streamlined process that is easy to use and enforce. When an IT team member creates a request for change, they are responsible for documenting the details that will help others understand what change needs to be implemented, and why they are making the request. The initial change request should

included details about what service and IT system is impacted, the expected risk, and the implementation details if known at the time of submission. The more information they include at the beginning, the more efficiently the request will move through the process. Here's some information that may be included in the request for change:

- Description of how the change will be implemented
 - Is this change request the result of any related incidents?
 - The impact that the change would have on all associated services and systems
 - Change impact, urgency, and risk assessment
 - Contact information for everyone involved in the change
 - Listing of initial change approvers (this list may be modified by the change manager)
 - Backup plan to follow in case the change is not successful
2. **Reviewing a Request for Change** - the change manager or their designated member is responsible for reviewing and evaluating all change requests submitted to the change management team. During this stage of the process they will evaluate the change request to determine if the request is reasonable and provide feedback to the requester if additional information is required. In some organizations a formal approval process by the change manager is put in place to provide a check-point before a change request moves onto the planning phase. During this review, the change will be evaluated according to the impact the change will have on the IT services, systems and business. The ultimate determination a change manager has to make at this point is the likelihood of a successful change implementation and the impact on the business.
 3. **Planning for a Request for Change** - once a change request moves into this phase it's ready for the detailed planning that is required to ensure it is successfully implemented. Some of the information documented in the change request includes the following:
 - A detailed change plan that outlines the course of action for implementing the change. The details should include the type of change, the priority associated with a change request, and the outcomes that could occur if the change is not made.
 - A complete listing of resources needed to complete the change.
 - Change implementation timeline - change manager will assist with choosing the appropriate dates based on a centralized change scheduling calendar they maintain that helps determine the best timing for when a change should occur.
 - Change testing plan - typically needed for major production changes where a successful test point is needed before implementing the change in production.
 - Change back out plan - for higher risk change requests a back out plan is needed so the IT team has a document to refer to if they need to quickly roll back the change.
 4. **Approving a Request for Change** - based on the type of change and risk the change manager will route it to the appropriate groups for approval. This may be a simple peer review or a more involved CAB approval for high risk production changes.
 5. **Implementing a Request for Change** - Implementing a change, especially in a production environment, is not a simple process. The change has to be constructed and prepared during the planning process, this is where most of the hard work is completed. Once the change has been implemented, tests must be done to determine whether the desired results have been achieved. If the change is not successful, troubleshooting may be used to determine what went wrong, and to implement a backup plan to alleviate the issues that necessitated the change request. It is very important that the change implementation assignee documents the results, whether the change is successful or needs to be rolled back. This information is valuable to help the team learn from any failures so they can improve their process and capabilities in the future.
 6. **Change closure and review** - The post-implementation review is an essential part of the change management process and a key aspect for improving the change management process. This part of the phase allows the change management team to understand whether your change procedures are working as effectively as possible. During this phase of the process the change manager will review a completed change request to determine whether the change was successful, whether the change timing was appropriate, and the expense of the change to determine the accuracy of estimates that were made during the planning phase. Reviewing change performance on a regularly basis provides an excellent check point to fine-tune your change management process for improvement.

Setup for change management in JIRA Service Desk

Configure the workflow and fields with the Change Management workflow add-on

We built the following workflow add-on based on the ITIL framework for change management: <https://marketplace.atlassian.com/plugins/com.atlassian.servicedesk.change/server/overview>.

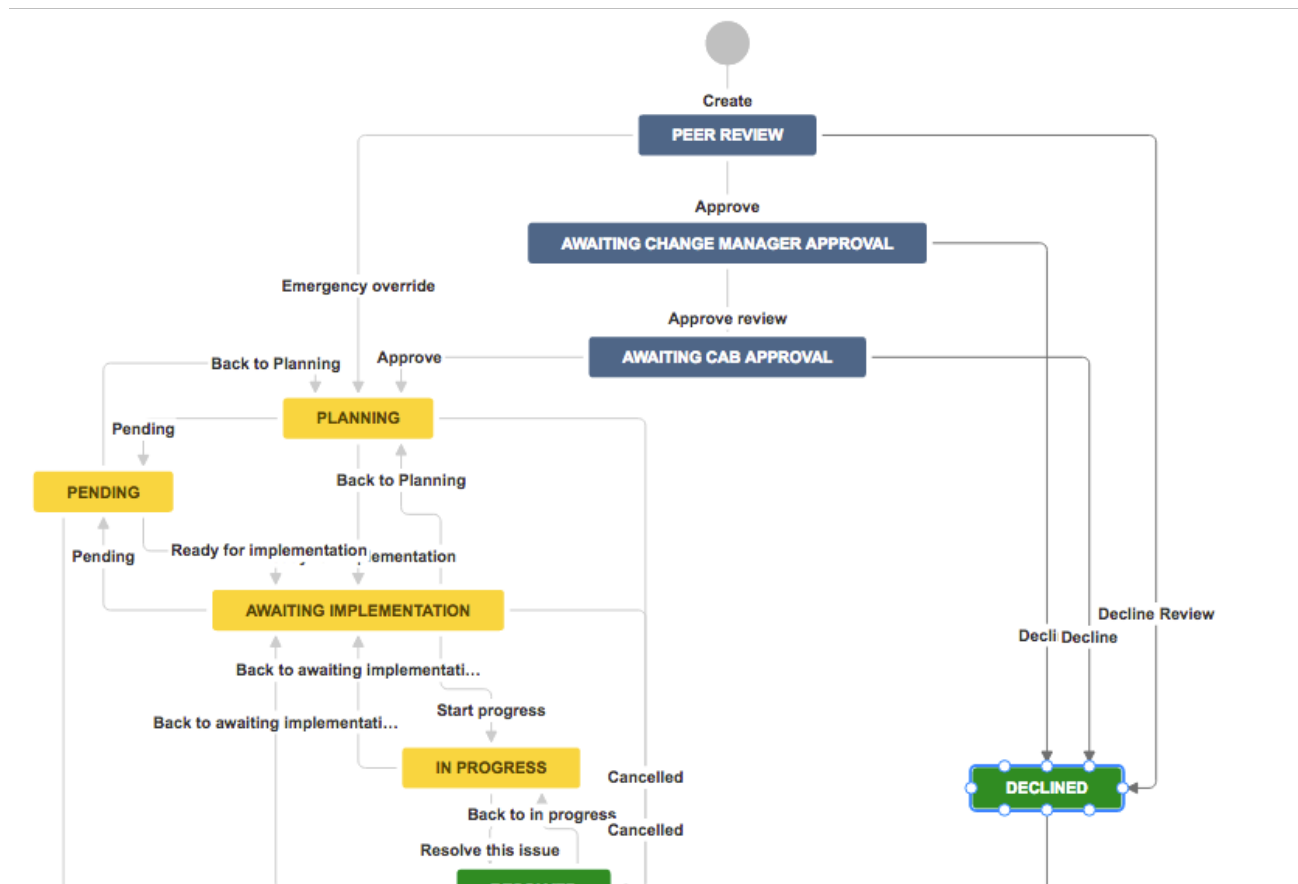
You can use this workflow as a basis that you can then build out your own change management process on.

To use the workflow from the Marketplace:

1. Log in as a user that has the JIRA administrator global permission, and follow the instructions listed here to [import a workflow](#).
2. To have the fields created by the workflow import displayed on your change issues, activate the screen by following the instructions here: <https://confluence.atlassian.com/adminjiracloud/defining-a-screen-776636475.html#Definingascreen-Activatingascreen>.

What does the workflow look like?

This sample workflow has the following transitions and statuses.



When you import it, the workflow will create a few screens and some custom fields for you as shown below:

Preview of import

Statuses

The following statuses will be created in your instance of JIRA.

- [AWAITING CHANGE MANAGER APPRO...](#)

Screens

The following screens will be created in your instance of JIRA.

Name	Description
JIRA Service Desk Pending Reason screen for Change Management	
JIRA Service Desk: Change Create Issue Screen - 2	
Resolve and cancel screen for change management	Resolve and cancel screen. Created by the Change Management for JIRA Service Desk workflow.
Workflow Screen - 2	This screen is used in the workflow and enables you to assign issues

Custom Fields

The following custom fields will be created in your instance of JIRA.

Name	Type	Description
CAB	User Picker (multiple users)	Choose multiple users from the user base via a popup picker window.
Change managers	User Picker (multiple users)	Choose multiple users from the user base via a popup picker window.
Change risk	Select List (single choice)	A single select list with a configurable list of options.
Change type	Select List (single choice)	A single select list with a configurable list of options.
Impact	Select List (single choice)	A single select list with a configurable list of options.
Operation categorisation	Select List (cascading)	Choose multiple values using two select lists.
Pending reason	Select List (single choice)	A single select list with a configurable list of options.
Product categorisation	Select List (cascading)	Choose multiple values using two select lists.
Reviewers	User Picker (multiple users)	Choose multiple users from the user base via a popup picker window.
Urgency	Select List (single choice)	A single select list with a configurable list of options.

*To display these fields for your

Change management fields

We recommend the use of the following fields to support your change management process:

- **Description:** Use this field to capture the summary of the change request
- **Business Justification:** Describes the business reason for implementing the change
- **Status:** Indicates the current state of the change request.
- **Pending Reason:** Specifies the reason for why a change request is moved into pending and what the

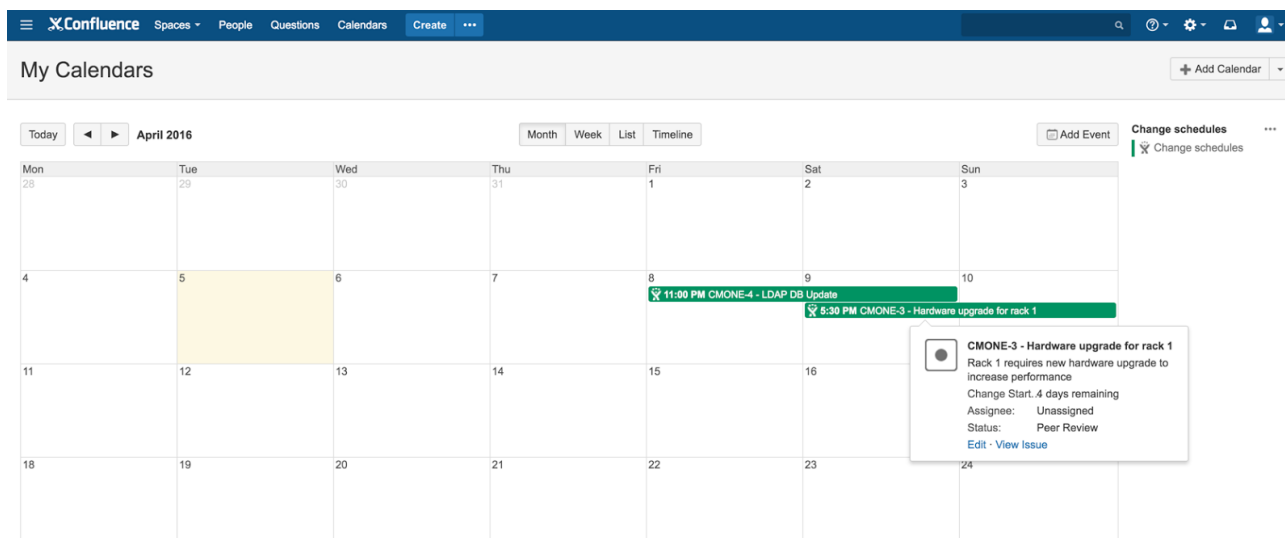
change request is pending on.

- **Example values:** Waiting on vendor, More info required, Awaiting approval, Pending on change request
- **Priority:** This is determined by the urgency and impact of the change. Your team can define the value matching according to your own processes.
 - Example values: Critical, High, Medium, Low
- **Urgency:** A measure how quickly a resolution of the change is required
 - Example values: Critical, High, Medium, Low
- **Impact:** A measure of the extent of the change and of the potential damage caused by the change before it can be resolved.
 - Example values: Extensive / Widespread, Significant / Large, Moderate / Limited, Minor / Localized
- **Operational categorization:** Classifies an change for the purpose of assignment and reporting from the operational perspective.
 - Example values: Configuration > Printer
- **Product categorization:** Classifies an change it for the purpose of assignment and reporting from the product perspective.
 - Example: Hardware > Printer
- **Change type:** Classifies the type of change to be implemented. Change types include Standard, Normal, Emergency.
- **Change reason:** indicates the reason for attempting the change. This field provides classification that is useful for reporting purposes.
 - Example: Repair, Upgrade, Maintenance, New Functionality, Other
- **Change risk:** Classifies the overall risk of implementing the change. The higher the number the higher the likelihood of a failed implementation. Various methods exist for determining the change risk based on an assessment of various factors which include: complexity, scope, testing, recovery and timing associated with the change. The change manager should review these factors to determine the appropriate change risk value to assign to the request.
 - Example: Critical High, High, Moderate, Low
- **Change Start Date:** identifies the start date for when the change will be implemented
- **Change Completion Date:** identifies the completion date for when the change should be completed
- **CAB Approval:** indicates the persons responsible for reviewing and approving a change request before implementation begins.
- **Component:** Indicates the service impacted by change
- **Resolution:** Classifies the reason for completing the change. This is referred to as Change Closure.

Change scheduling

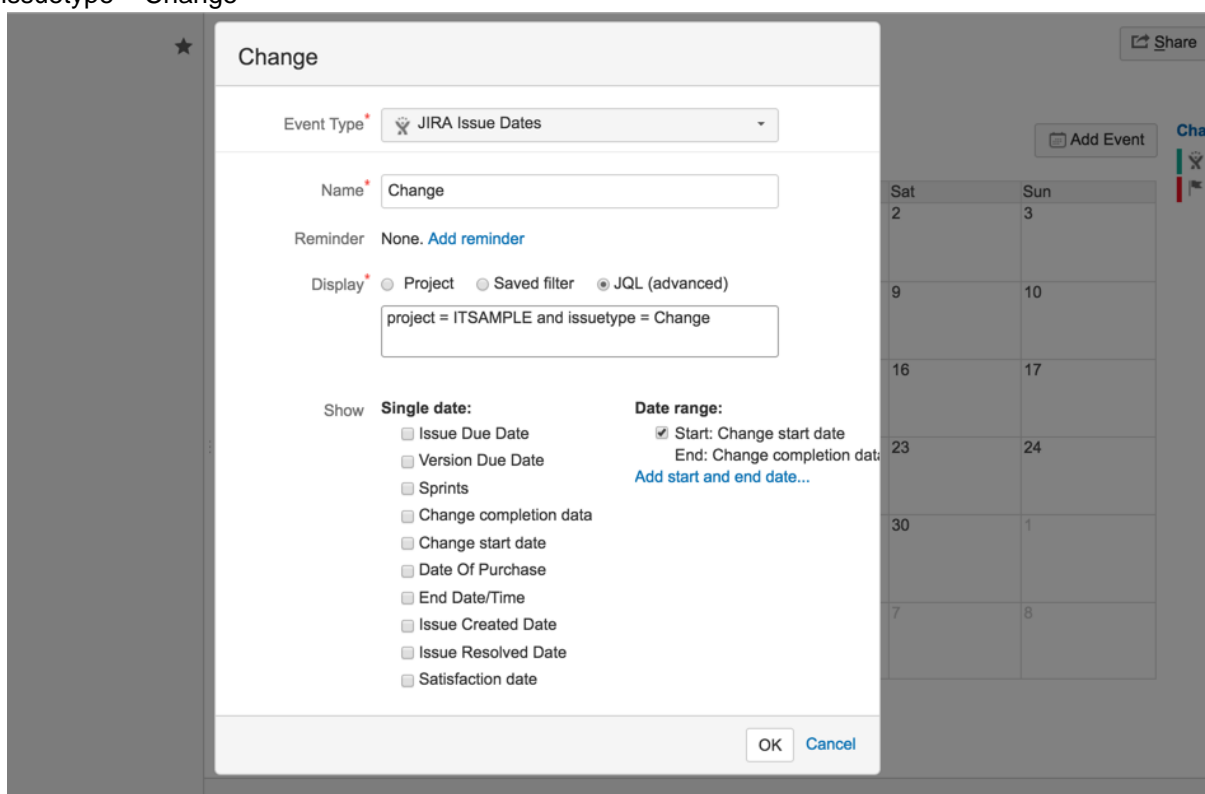
A change schedule (sometimes known as a change calendar) gives you an clear graphic view on past and future change requests. This calendar view of change requests allows you to co-ordinate your change events with major business events and/or scheduled releases.

With [Team Calendars for Confluence](#), you can visualize and track change request raised from JIRA Service Desk on the calendar view. It's easy to see the full details for change requests from the calendar view by going to the issues themselves.



To set up the change calendar:

1. Add [Team Calendars for Confluence](#) to your site.
2. Go to the Confluence and choose the space for your team, e.g. 'IT Sample'.
3. In the left sidebar, click **Calendars**.
4. Select **Add Calendar** (top right).
5. After a Calendar is added, select **Add Event**.
6. In the Add Event dialog, on Event type dropdown select **JIRA Issue Dates**.
7. For Display, select JQL "advanced". In the JQL type in: project = "Your IT project name" AND issuetype = Change



8. A list of dates should show up, select "Add start and end date..." under **Date range** and select **Change start date** as the start date and **Change completion date** as the end date.
Note: these two fields are added by the **Change Management for JIRA Service Desk** workflow mentioned above. If you have your own fields for the dates, use your own fields in this step.
9. After clicking **Add** and then **OK**, you now have a change schedule!

From now on, whenever a change request is raised in the project, the change schedule will automatically display it on this calendar.

Problem management

What is problem management?

While incident management is all about finding the shortest path to restoring normal service, problem management takes the long view with the primary goal of finding the underlying causes of an incident, and the best resolution and prevention. The aim of problem management is to reduce the adverse impact of incidents that are caused by errors within the IT infrastructure, and to prevent recurrence of incidents related to these errors. Problem investigations should be addressed with a prioritized approach that takes on problems that have the greatest potential for causing serious disruption to critical IT services.

When incidents occur, the responsibility of the incident response team is to restore normal service as quickly as possible, without necessarily identifying or resolving the underlying cause of the incident. If incidents occur rarely or have little impact, assigning resources to perform root cause analysis is difficult to justify. However, if a major service outage incident or a series of repeated incidents cause significant impact, the problem management team is tasked with investigating the underlying cause of the incidents, and to identify the best method to eliminate the root cause.

This page

- [What is problem management?](#)
- [Benefits](#)
- [Problem management process](#)
 - [Problem Management process summary](#)
- [Setup for problem management in JIRA Service Desk](#)
 - [Configure the workflow and fields](#)
 - [Fields](#)
- [Publishing a Known Error to the Confluence IT space](#)

Related pages

- [Incident management](#)
- [Change management](#)
- [Service request fulfillment](#)

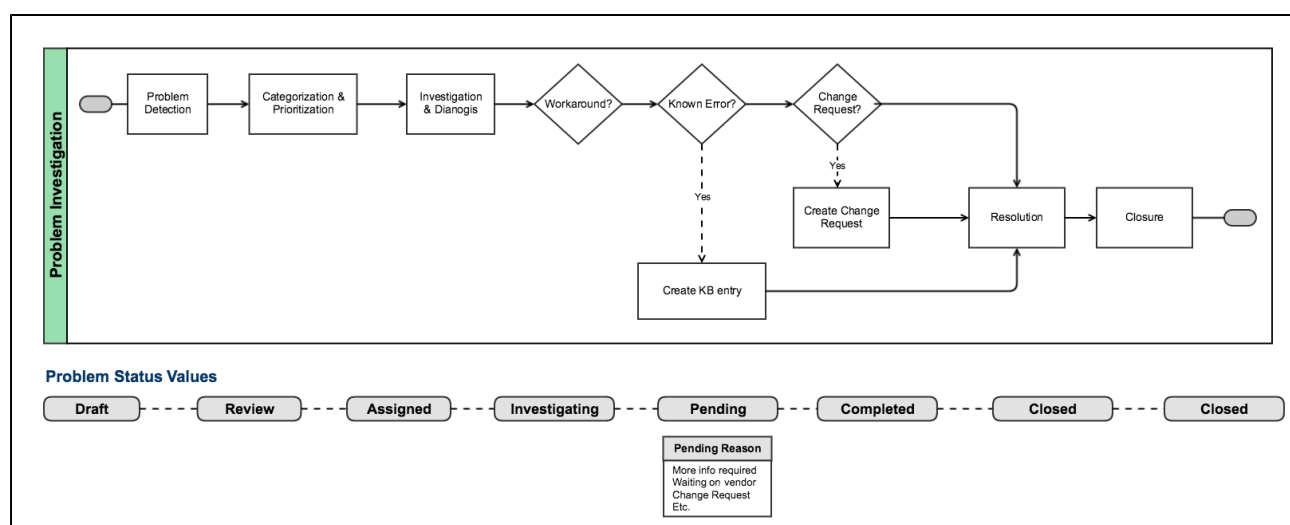
Benefits

Problem management provides a logical extension of incident management for improving the overall quality of IT services. Some of the benefits for implementing a formal approach to problem management include:

- Improved quality of IT service which results in reliable services for the business
- Reduced incident volume that minimizes service interruptions for the business
- Improved sharing of knowledge across the IT organization as the team learns from past mistakes and issues.
- Valuable reporting and analytic insights from historical data to identify trends and the means of preventing failures or reducing the impact of failures.
- Permanent solutions that help reduce the number and impact of incidents over the long term.
- Improved first touch resolution by the service desk team because they have access to lessons learned, known errors and work arounds documented in a central knowledge base.

Problem management process

This process represents a example of problem investigation based on ITIL recommendations. It provides a streamlined example that most customers can use as a starting point to adapt their existing ITIL processes and support processes.



Problem management works by using analysis techniques to identify the cause of the problem. Incident management is not usually concerned with the cause, only the cure: restoration of service. Problem management, therefore, takes longer and should be done once the urgency of the incident has been resolved. A problem investigation is often a direct result of a Post Incident Review (PIR) where the IT team needs to know the root cause for a recurring service outage. Problem management can take time. It is important to set time limits or the cost of resolution can become expensive.

The first activity of a problem investigation is to diagnose the problem and validate any workarounds. During this process, it's important that the IT team involved in the investigation document their findings and any workarounds they identify. Once the problem has been diagnosed and a workaround identified, the problem is referred to as a "known error." These are documented in a centralized knowledge base referred to as the known error database (KEDB). The knowledge shared in the known error database is a significant resource for the service desk team responding to new incidents.

Problem Management process summary

The ITIL problem management process includes many vital steps to ensuring a problem investigation is successfully completed. Here's a summary of the steps included in a problem investigation:

1. **Detect the problem** - The service desk will typically raise a problem because they are seeing incidents occur across the organization with similar conditions, or they are seeing a reoccurring incident. The problem investigation team is responsible for reviewing these type of problems when they are reported. They are also responsible for a proactive review of incident patterns and alerts from event management to identify patterns they should investigate that may have an impact on overall quality of IT services.
2. **Create a problem record** - When a potential problem is detected it should be recorded by either the service desk team or the problem management team. A problem record should capture the time and

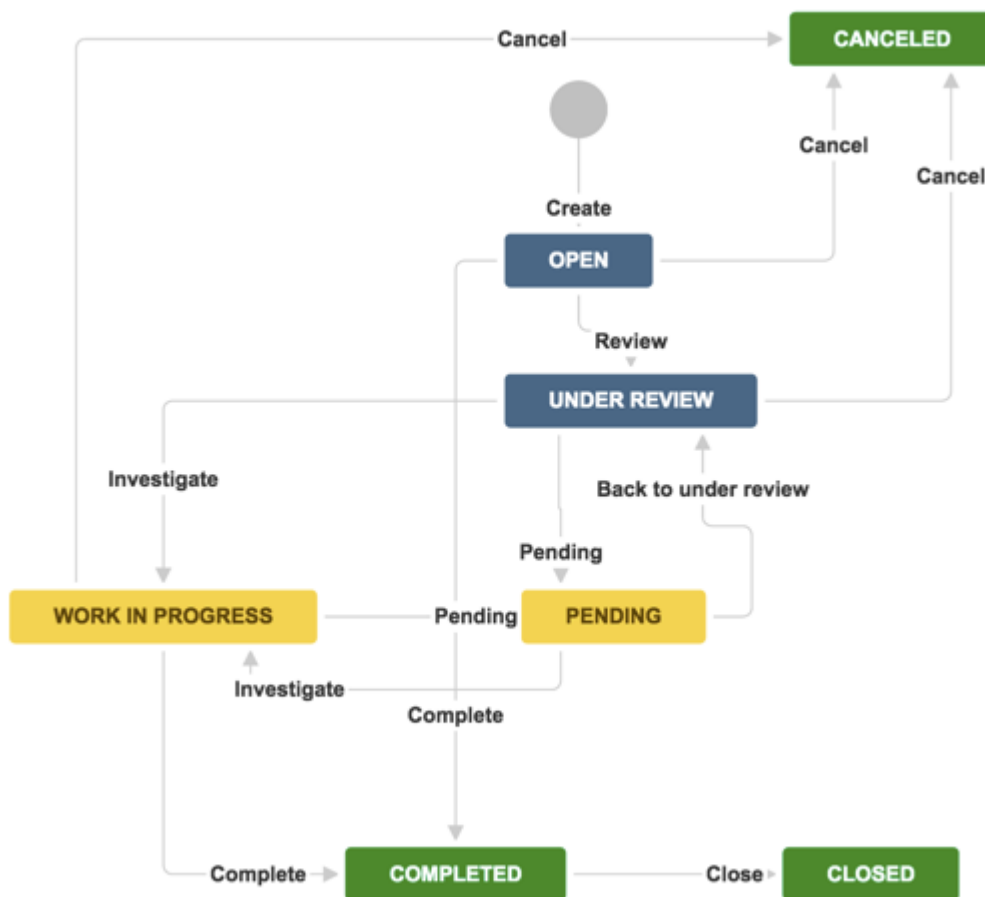
date of occurrence, the symptoms, all related incident(s), any previous troubleshooting steps, and the problem category and source. This information helps the problem management team research the root cause.

3. **Categorize the problem** - When a problem is recorded the categorization should match incident categorization. The use of incident and problem categorization allows the service desk to sort and model incidents that occur regularly. The use of categorization for of these type of records allows the IT organization to track problem trends and assess the impact of service capacity, demand and quality.
4. **Determine a problem's priority** - The overall priority of a problem is determined by the impact on users and the business. Urgency is another important aspect that the problem investigation team need to consider. Urgency is how quickly the organization requires a resolution to the problem. The impact is a measure of the extent of potential damage the problem can cause to services and the organization. Prioritizing the problem allows an IT team to utilize investigative resources most effectively.
5. **Investigating and diagnosing the problem** - The speed at which a problem is investigated and diagnosed depends on its assigned priority. High priority problems should be addressed first since their impact on services and the organization are the greatest. Having the proper problem classification and categorization helps drive efficiency in the problem management process. It helps the problem investigation team identify trends more quickly. Once the problem team starts working on a problem they quickly move into the diagnosis phase of the process. This typically involves analyzing the incidents that lead to the problem along with any previous troubleshooting steps the service desk team took to find a workaround. This level of analysis typically involves detailed review of data and conferring with experts to gain the proper insights needed to find the root cause of the issue.
6. **Identify a workaround for the problem** - In most cases when a problem investigation is in progress, incidents remain open. The primary goal in these cases is to find an appropriate workaround that will enable the service desk to restore normal service. Even after the incident is resolved with a workaround, the problem investigation continues. Problem investigations can be lengthy and take considerable time to resolve, therefore a workaround is vital. A workaround should only be considered a temporary measure.
7. **Create a known error record** - Once the workaround has been identified it should be shared with the service desk team as a known error. It's good practice to record a known error in a knowledge base (KB) article. ITIL refers to this KB as a known error database (KEDB). Documenting the workaround allows the service desk team to resolve incidents quickly and avoid further problems being raised on the same issue.
8. **Resolve the problem** - Problem investigations are considered resolved when they find the underlying root cause of a set of incidents and prevents those incidents from recurring. Some resolutions may require a change request to implement the final resolution. An example of this is when a software update is required to a production application to address a know performance issue. The final important part of the resolution phase for a problem is documenting the steps taken to implement the resolution. This should be published to the IT knowledge base so the service desk has the information for future reference.
9. **Final review of the problem** - ITIL classifies this as a major problem review. It's an important step that many IT organizations skip. A major problem review is an important activity that helps prevent future problems. During a review, the problem management team evaluates the problem investigation documentation to identify what happened and why. This allows the team to identify important lessons learned that help identify any process improvements that are needed. Lessons learned during this step should be documented and shared with the IT organization.

Setup for problem management in JIRA Service Desk

Configure the workflow and fields

We recommend the following workflow for managing problem records.



Fields

We recommend the use of the following fields for your problem management process:

- **Description:** Use this field to capture the basics about the problem.
- **Status:** Indicates the current state of the problem.
- **Pending Reason:** Specifies the reason for why problems are moved into pending and what the incident is pending on.
 - **Example values:** Waiting on vendor, More info required, Awaiting approval, Pending on change request.
- **Priority:** This is determined by the urgency and impact of the problem. Your team can define the value matching according to your own processes.
 - Example values: Critical, High, Medium, Low
- **Urgency:** A measure how quickly a resolution of the problem is required
 - Example values: Critical, High, Medium, Low
- **Impact:** A measure of the extent of the problem and of the potential damage caused by the Incident before it can be resolved.
 - Example values: Extensive / Widespread, Significant / Large, Moderate / Limited, Minor / Localized
- **Operational categorization:** Classifies a problem for the purpose of assignment and reporting from the operational perspective.
 - Example values: Configuration > Printer
- **Product categorization:** Classifies a problem for the purpose of assignment and reporting from the product perspective.
 - Example: Hardware > Printer
- **Source:** Indicates where the problem comes.
 - Example values: Phone, Email, Monitoring event
- **Component:** Indicates the service impacted by problem

- **Resolution:** Classifies the resolution of the problem, e.g. Known error.
- **Root cause:** Once the problem investigation is completed and the root cause is identified, it's good practice to document the result. The root cause field allows you to document the details of the root cause. Depending on the results of the problem investigation you may also need to publish the root cause as a known error to the knowledge base.
- **Workaround:** Documents the temporary solutions when the final solution is not available or implemented yet.

Publishing a Known Error to the Confluence IT space

Once a problem investigation has been diagnosed and a workaround identified, the problem is referred to as a “known error.” These results of these type of problem investigations should be published to the known error database. Confluence provides an ideal location for publishing these type of knowledge articles that will provide significant resource for incident management when resolving incidents caused by known errors. Once the known error has been identified, the next step is to determine how to fix it. This will typically involve a change request for the impacted system.

The following is an example of a Known Error Data Base (KEDB) published in Confluence.

The screenshot shows a Confluence page within the 'IT Support' space. The page title is 'Webstore MS IIS Performance issue - 234234', created by Paul Buffington on April 28, 2016. The content is a Microsoft Security Bulletin MS16-049 - Important Security Update for HTTP.sys (3148795), published on April 12, 2016. The page version is 1.0. The 'On this page' section lists: Executive Summary, Affected Software and Vulnerability Severity Ratings, Vulnerability Information, Security Update Deployment, Acknowledgements, Disclaimer, and Revisions. The 'Executive Summary' section states that the security update resolves a vulnerability in Microsoft Windows that could allow denial of service. The 'Affected Software and Vulnerability Severity Ratings' section lists affected software versions and provides a link to the Microsoft Support Lifecycle for more information.

Incident management

What is incident management?

The service desk team typically provides the first line of response for incidents since they provide a single point of contact for customer communications with IT. According to ITIL, the aim of incident management is to restore any disruption or failure of service to normal operations as quickly as possible. This includes monitoring for any condition that has the potential to result in a reduced quality of service. How an IT team responds to incidents is one of the top priorities for many IT organizations. ITIL provides best practice guidance for defining a proactive approach to incident management.

The service desk team typically provides first line of support (level 1) when responding to an incident. Activities performed by the service desk team include the following:

- Record the incident - date & time, description, name of person reporting it and unique identification number
- Incident identification, classification and prioritization
- Initial diagnosis
- Communication with the customers and service owner throughout the

This page

- life of the incident
- Escalation to level 2 support (as needed)
- Resolution and verification
- Incident closure
- Participate in a Post Incident Review (PIR) for all major incidents.

- What is incident management?
- Incident management process
- Setup for incident management in JIRA Service Desk
 - Configure the workflow and fields with the Incident Management workflow add-on
 - What does the workflow look like?
 - Incident management fields

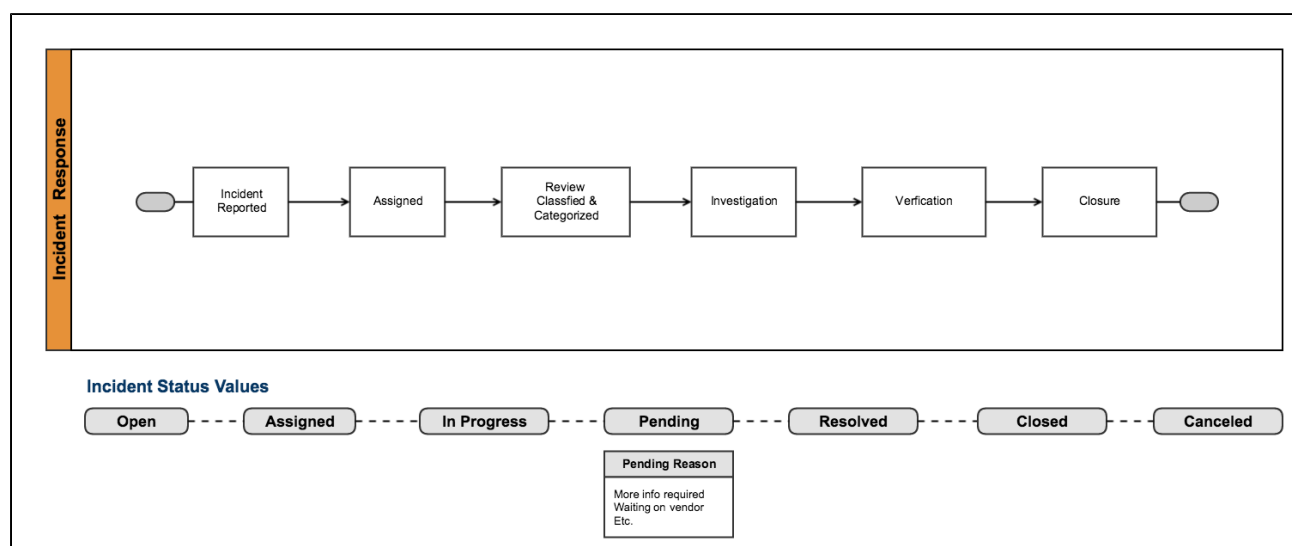
Related pages

- Problem management
- Change management
- Service request fulfillment

Since the goal of incident management is to restore a service as quickly as possible, the service desk is not expected to perform root cause analysis to identify *why* an incident occurred. Root cause analysis is the focus of a problem investigation, and is covered in [Problem management](#). However, it's important that the service desk team capture all relevant information while working on the issue, because this may be important when responding to future incidents.

Incident management process

It's important that the service desk team use all available resources to effectively respond to an incident. This includes following a predefined process to streamline the response and reduce the risk of prolonged service outages. The following process represents an example incident response based on ITIL recommendations. It provides a streamlined starting point that you can use for adapting existing ITIL processes or defining new ones.



Variations in this process may depend on your IT organizations' process and work instructions. Common incident response process considerations include:

- What Service Level Agreements (SLAs) are required that define incident priorities, escalation paths, and resolution time frames?
- What incident procedures are needed to provide standardized responses to ensure incidents are resolved efficiently?
- What types of incident categorizations are required for better data gathering and problem management?
- What is needed for incident statuses, categories, and priorities to properly classify, track and report on incidents?
- What is the process and procedure for major incident response?
- What are the proper incident management role responsibilities and assignments needed to ensure an effective process?

Setup for incident management in JIRA Service Desk

Configure the workflow and fields with the Incident Management workflow add-on

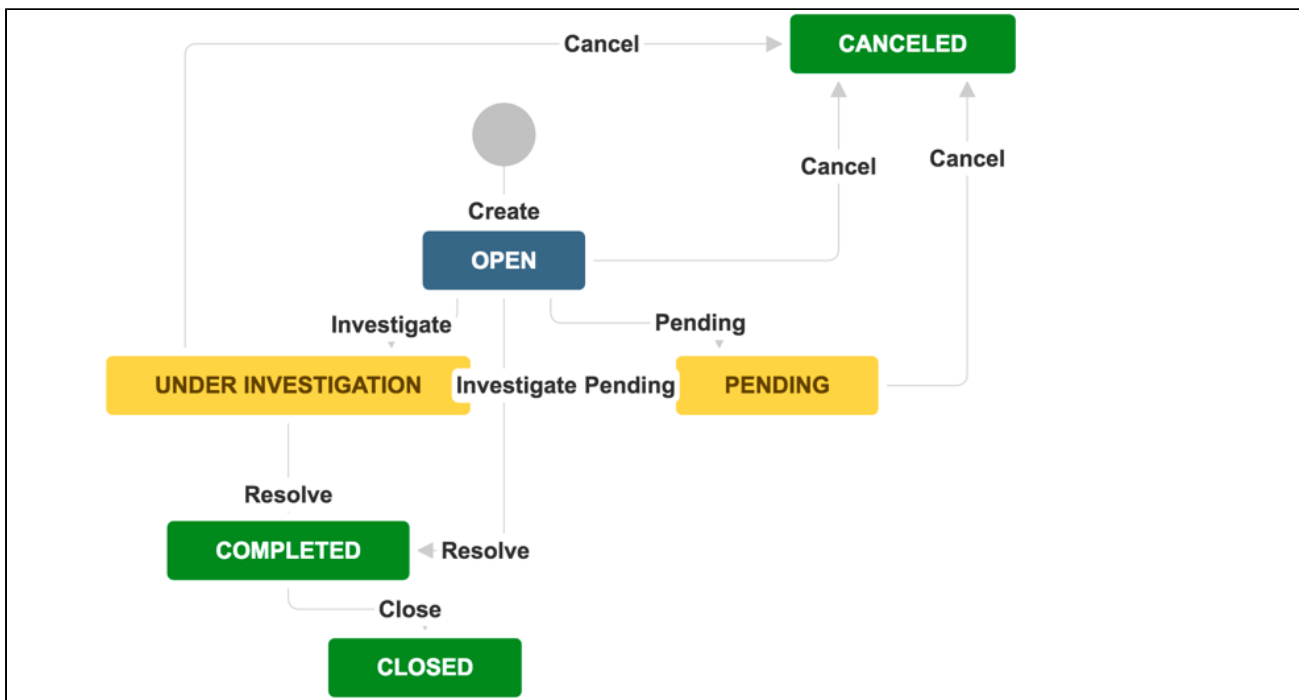
We built the following workflow add-on based on the ITIL framework for incident management: <https://marketplace.atlassian.com/plugins/com.atlassian.servicedesk.incident/server/overview>.

You can use this workflow as a basis that you can then build out your own change management process on.

To use the workflow from the Marketplace:

1. Log in as a user that has the JIRA administrator global permission, and follow the instructions listed here to [import a workflow](#).
2. To have the fields created by the workflow import displayed on your incidents, activate the screen by following the instructions here: <https://confluence.atlassian.com/adminjiracloud/defining-a-screen-776636475.html#Definingascreen-Activatingascreen>.

What does the workflow look like?



When you import it, the workflow will create a few screens and custom fields as shown below:

Preview of import

Statuses

The following statuses will be created in your instance of JIRA.

- **IN PROGRESS**

Screens

The following screens will be created in your instance of JIRA.

Name	Description
JIRA Service Desk Pending Reason screen for Incident Management - 2	Screen to specify the pending reason when transitioning to the Pending status
JIRA Service Desk: Incident Create Issue Screen - 2	
Resolve Issue Screen - 2 - 16	Allows to set resolution, change fix versions and assign an issue.
Workflow Screen - 3	This screen is used in the workflow and enables you to assign issues

Custom Fields

The following custom fields will be created in your instance of JIRA.

Name	Type	Description
Group assignee	Group Picker (multiple groups)	Choose multiple user groups using a popup picker window.
Impact	Select List (single choice)	A single select list with a configurable list of options.
Pending reason	Select List (single choice)	A single select list with a configurable list of options.
Urgency	Select List (single choice)	A single select list with a configurable list of options.

Incident management fields

We recommend the use of the following fields to support your incident management process:

- **Description:** Use this field to capture the symptoms and the basic information about the incident.
- **Status:** Indicates the current state of the incident.
- **Pending Reason:** Specifies the reason for why incidents are moved into pending and what the incident is pending on.
 - **Example values:** Waiting on vendor, More info required, Awaiting approval, Pending on change request, Pending on problem investigation.
- **Priority:** This is determined by the urgency and impact of the incident. Your team can define the value matching according to your own processes.
 - Example values: Critical, High, Medium, Low
- **Urgency:** A measure how quickly a resolution of the Incident is required
 - Example values: Critical, High, Medium, Low
- **Impact:** A measure of the extent of the Incident and of the potential damage caused by the Incident before it can be resolved.
 - Example values: Extensive / Widespread, Significant / Large, Moderate / Limited, Minor / Localized

- **Operational categorization:** Classifies an incident for the purpose of assignment and reporting from the operational perspective.
 - Example values: Configuration > Printer
- **Product categorization:** Classifies an incident for the purpose of assignment and reporting from the product perspective.
 - Example: Hardware > Printer
- **Source:** Indicates where the incident comes.
 - Example values: Phone, Email, Monitoring event
- **Component:** Indicates the service impacted by Incident
- **Resolution:** Classifies the resolution of the incident.

Automating the calculation of priority based on impact and urgency values on issue creation

Some IT teams use an urgency-impact matrix to determine the priority of an issue. In this type of process, automatically setting the priority according to the urgency and impact values at the creation of an issue becomes handy because it helps with issue triage, for example, you can set up your service desk to have different queues for incidents of different priorities and incidents with the corresponding priority will be put into the correct queue as soon as they are created. The auto-setting of the priority value at issue creation time also helps ensure that the appropriate SLAs are applied right from the beginning of an issue.

You can set up the automation for this with workflow post functions and automation rules.

Example

By the end of this tutorial, you will automate the priority calculation after a change request is submitted according to the following matrix.

Table: Priority values

	Urgency			
Impact	Critical	High	Medium	Low
Extensive	Critical	Critical	High	Medium
Significant	Critical	High	Medium	Medium
Moderate	High	Medium	Medium	Low
Minor	Medium	Medium	Low	Low

Before you begin

Make sure you have the following two [custom fields](#) of the type **Select List (single choice)**, each containing the values listed the previous table:

- **Urgency:** Critical, High, Medium, Low
- **Impact:** Extensive, Significant, Moderate, Minor

Automating the calculation

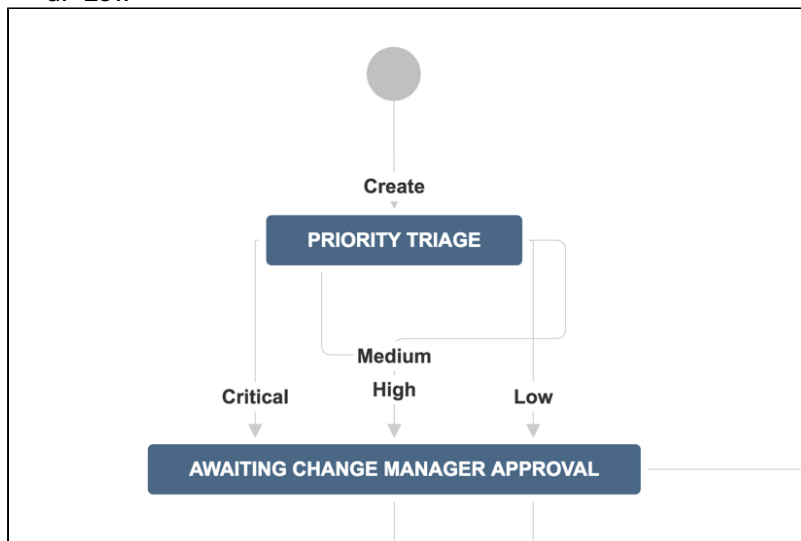
After step 1, the workflow will transition a change to the **Priority triage** status as soon as an issue is created. You will also have 4 transitions to set the **Priority** to a value.

With step 2, you will set up an automation rule that checks to the value of Urgency and Impact fields and fires off the corresponding transition according to the matrix.

Step 1: Configuring the workflow

1. Go to the workflow that is used by your **Change** issue type.

2. Between the **Create** transition and your first status, add a new status and name it **Priority Triage**.
3. Add the following four transitions from this status:
 - a. Critical
 - b. High
 - c. Medium
 - d. Low



4. In each of these transitions, [add a post function](#).
 - a. Select the **Update Issue Field** post function.
 - b. In the post function, update the **Priority** field to match the transition, for example, the Critical transition will have a post function that changes the **Priority** field to **Critical**. Similarly, the **High** transition will have this post function to set the **Priority** field as **High**, and the **Medium** transition to set the value to **Medium**, and **Low** to **Low**.

Workflows / Change Management with priority triage for Jira Service Desk (Draft)

Transition: Critical

PRIORITY TRIAGE Critical

Screen: None - it will happen instantly

Triggers 0 Conditions 0 Validators 0 Post Functions 6

The following will be processed after the transition occurs

1. The **Priority** of the issue will be set to **Critical**. ←
2. Set issue status to the linked status of the destination workflow step.
3. Add a comment to an issue if one is entered during a transition.
4. Update change history for an issue and store the issue in the database.
5. Re-index an issue to keep indexes in sync with the database.
6. Fire a **Generic Event** event that can be processed by the listeners.

5. Publish the workflow.

Step 2: Configuring the automation rule

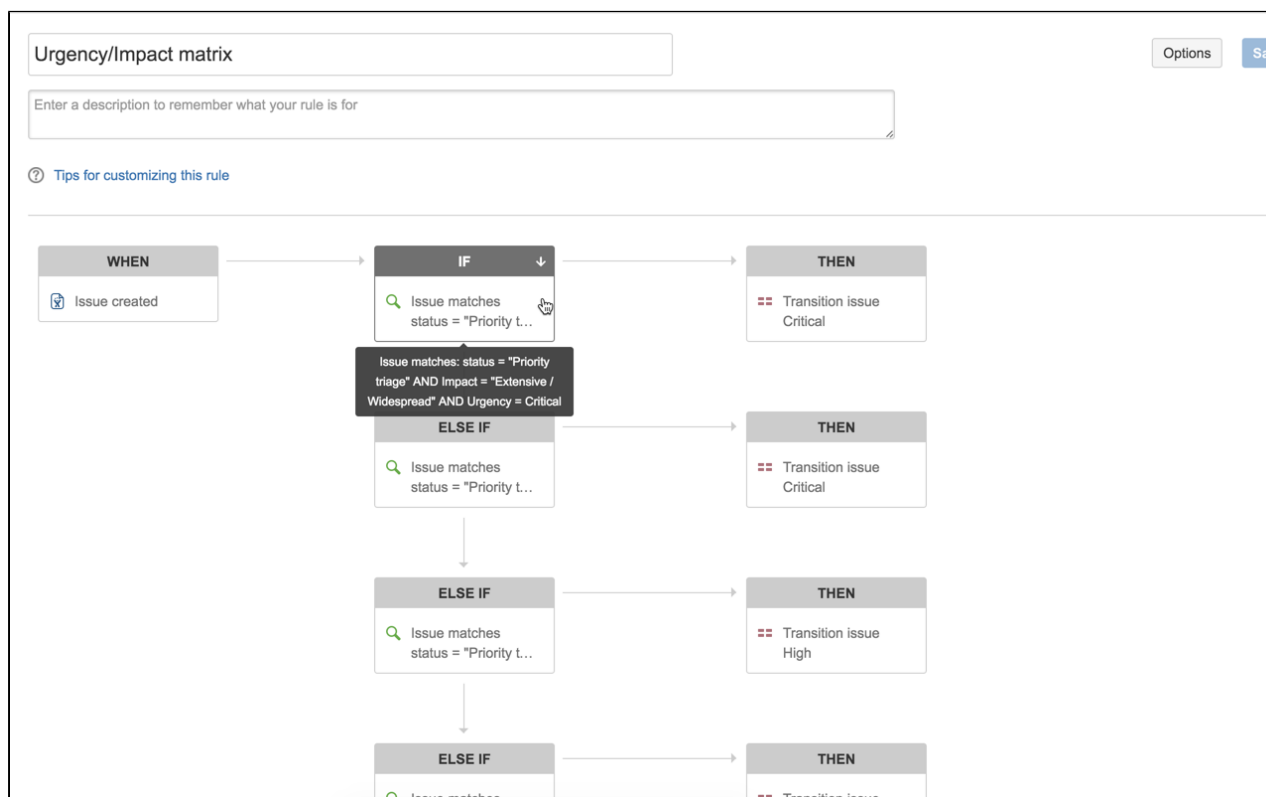
After the post function is set, let's create an automation rule that triggers the appropriate transition depending on the urgency and impact selected during request creation.

The rules should follow the following pattern:

- **When:** Issue created
- **If, or Else if:** Specify the urgency and impact value pair according to the matrix, e.g.,

```
status = "Priority triage" AND Impact = "Extensive / Widespread"
AND Urgency = Critical
```

- **Then:** Transition issue, and select the transition that matches the value pair according to the matrix.



Note: If your **Urgency** or **Impact** value is optional on the request type form or issue create screen, then there might be cases where these fields are empty. In this case, make sure that you add a **Else if** condition that caters for this scenario. For example,

- **Else if:**

```
status = "Priority triage" AND Impact is empty OR Urgency is EMPTY
```

- **Then:** Transition issue, **Low**.

Getting help with JIRA Service Desk

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.

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). (<https://jira.atlassian.com>)

I don't like the way something works

Raise a [suggestion](https://jira.atlassian.com). (<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](#).

Tip:

If you are a JIRA system administrator, you have a number of additional support tools available. These include the ability to raise a support request from within your JIRA applications, create zip files of key JIRA application information, and more. For details, see the [Administering JIRA Applications](#) documentation.

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 instance or an unsupported platform,
- you are trying to perform an unsupported operation, or
- you are developing an add-on for JIRA Service Desk.

You can also have a look at the [most popular JIRA Service Desk answers](#).

JIRA Service Desk 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.