Structure by Tempo

Visualize, track and manage Jira projects

Exported on 04/15/2024

Table of Contents

1 Structure – Project Management at Scale

Structure for Jira is a project and portfolio management tool that's just as indispensable for small teams as it is for large enterprises.

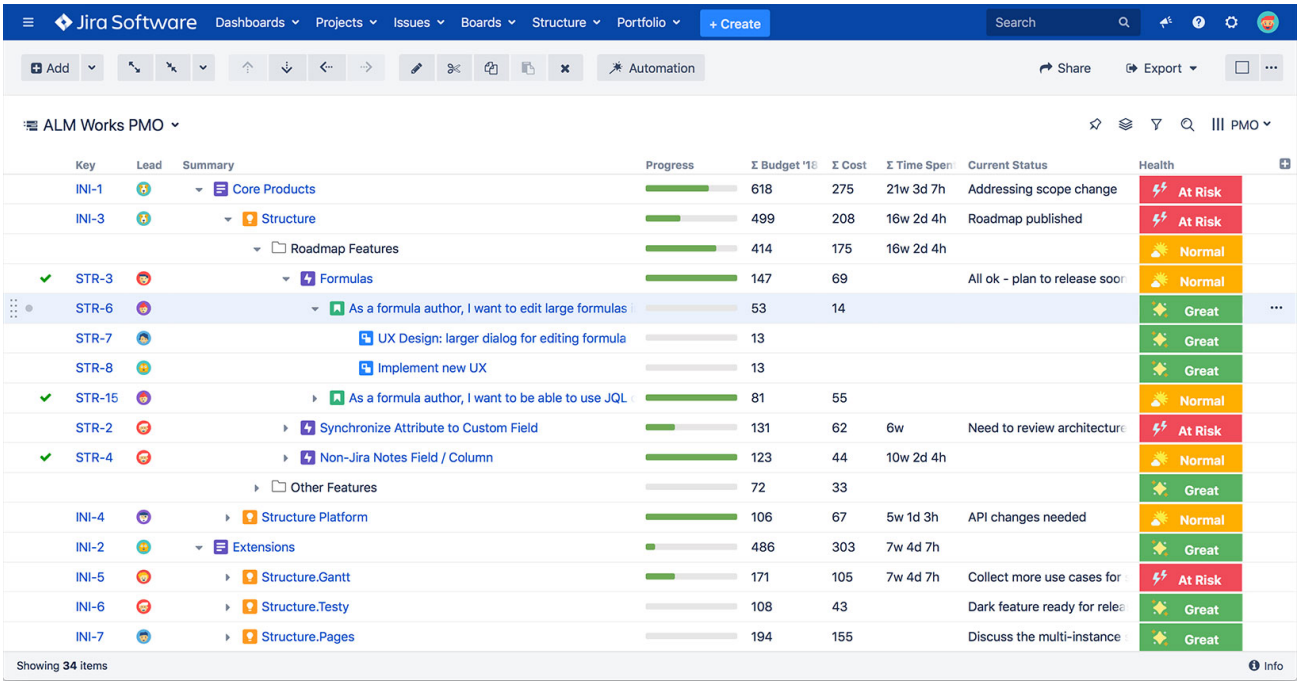
Structure lets you create and manage hierarchical lists, called **structures.** A structure may contain Jira issues, folders and other types of items. The hierarchy depth is not limited, and you can bring together issues and other information from multiple projects across company's portfolio.

Structure has a number of powerful features:

- Multiple, shareable structures
- Customizable views
- Sorting, filtering and search, including hierarchy-driven search
- Totals, progress calculations
- Automation – building dynamic structures that update in real time
- Mass cloning capability for template projects
- Powerful user interface with keyboard shortcuts
- Synchronization with issue links, agile boards, Jira sub-tasks
- Export to Excel
- On-the-spot issue creation and editing

2 Getting Started with Structure

Structure[1] allows you to visualize, track and manage progress across Jira projects and teams, using adaptable, user-defined issue hierarchies presented in a familiar spreadsheet-like view of Jira issues. In addition, structures may contain folders and other helpful organizational elements not found in Jira.



Key	Lead	Summary	Progress	Σ Budget '18	Σ Cost	Σ Time Spent	Current Status	Health
INI-1		Core Products		618	275	21w 3d 7h	Addressing scope change	At Risk
INI-3		Structure		499	208	16w 2d 4h	Roadmap published	At Risk
		Roadmap Features		414	175	16w 2d 4h		Normal
STR-3		Formulas		147	69		All ok - plan to release soor	Normal
STR-6		As a formula author, I want to edit large formulas		53	14			Great
STR-7		UX Design: larger dialog for editing formula		13				Great
STR-8		Implement new UX		13				Great
STR-15		As a formula author, I want to be able to use JQL		81	55			Normal
STR-2		Synchronize Attribute to Custom Field		131	62	6w	Need to review architecture	At Risk
STR-4		Non-Jira Notes Field / Column		123	44	10w 2d 4h		Normal
		Other Features		72	33			Great
INI-4		Structure Platform		106	67	5w 1d 3h	API changes needed	Normal
INI-2		Extensions		486	303	7w 4d 7h		Great
INI-5		Structure.Gantt		171	105	7w 4d 7h	Collect more use cases for	At Risk
INI-6		Structure.Testy		108	43		Dark feature ready for relea	Great
INI-7		Structure.Pages		194	155		Discuss the multi-instance	Great

The following guide will walk you through the basics of working with Structure, help you create your first structures and prepare you to build your own set of customized structures for tracking and analyzing projects across your organization.

2.1 Before You Begin

Before we get started, let's go over a few basic concepts, so the rest of this guide will make more sense:

- Structure lets you create 'containers' (called *structures* – with a lowercase 's') where you can add issues and arrange them into a meaningful hierarchy
- You can add issues from any number of Jira projects and arrange them in any way, regardless of issue type, status or any other properties
- You can create as many levels of hierarchy as you need

And one last note before we begin. This guide is only intended to cover the essential information for getting started and working with Structure. For an in-depth discussion of the many capabilities and features Structure has to offer, please refer to our Structure User's Guide(see page 82).

2.1.1 Let's Get Started

- Creating Your First Structure(see page 57)

1 https://hubs.la/Q02km0z10

- Working with Structure(see page 68)
- Getting the Most Out of Structure(see page 78)
- Help and Support(see page 80)

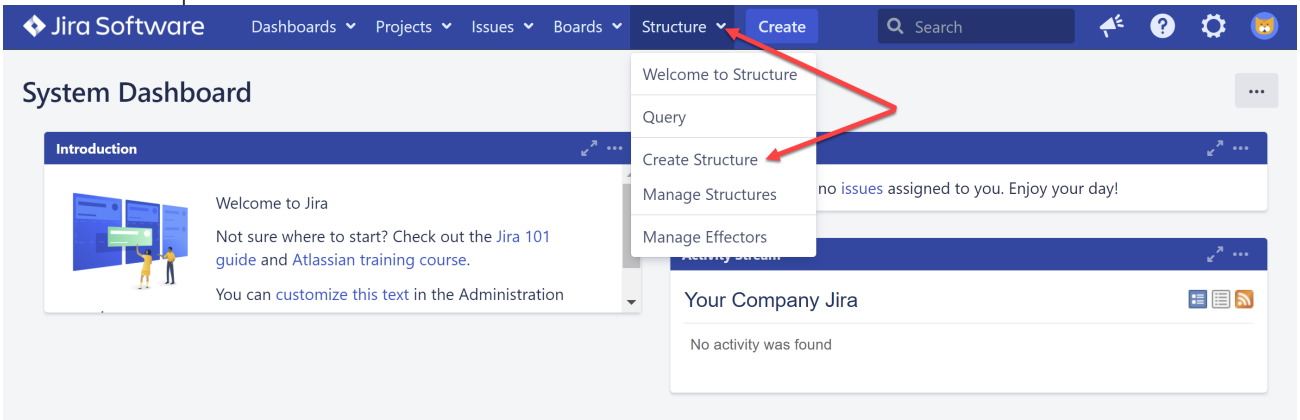
2.2 Creating Your First Structure

While there are endless ways to build a structure, we're going to show you some of our most popular (and useful) approaches, including our powerful automation tools(see page 58) and steps for adding issues manually[2].

We encourage you to experiment with the tools presented here, make several new structures and find some templates that will work well for you and your team

2.2.1 Your First Structure

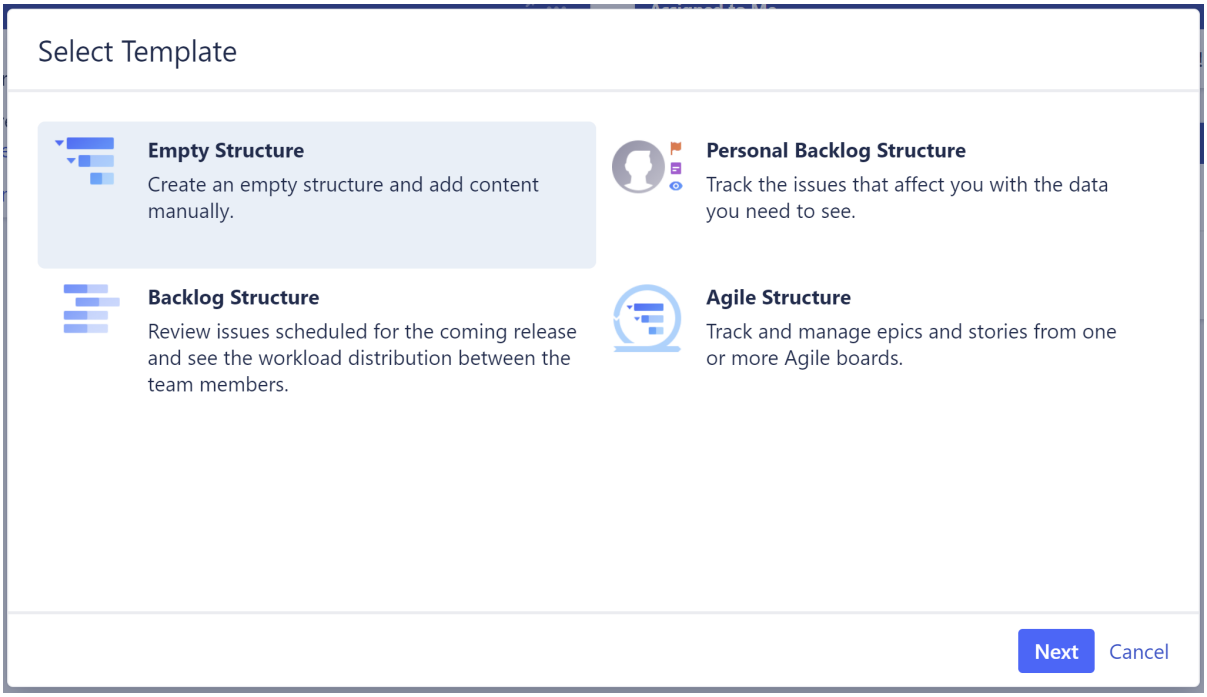
Regardless of how you plan to build your structure, you first need to create it. To do so, go to the top menu and select Structure | Create Structure.



You can also try our **Welcome to Structure** page - it contains several ideas and tutorials for building your own custom structures.

You have the option of using one of our template wizards to streamline the creation of your new structure, or starting with an empty structure. We encourage you to try our templates later, but for now let's select "Empty Structure."

2 https://wiki.almworks.com/display/structure/.Building+a+Structure+Manually+v8.2



Give your structure a name and select who it should be shared with (for more information, see Structure Permissions(see page 543)). When you're finished, click **Create**.

2.2.2 Next Steps

When the structure opens, it will be completely empty. In the next sections, we'll show you how to add and organize issues to a structure, using Automation. We will also discuss manually adding issues to a structure, but this is an advanced topic – so if you want to keep things simple, feel free to skip that part!

- Building a Structure with Automation(see page 58)

2.2.3 Building a Structure with Automation

Automation is a powerful feature that lets you create **dynamic structures**, which will update themselves when there are changes in Jira (and can update Jira when you make changes in the structure).

You can use Automation to build part or all of a structure. For the purpose of this guide, we're going to do the latter – both because we think it's useful and we want to show off how easy it is to create highly-specialized hierarchies using Structure (we're very proud of that!).

2.2.3.1 A (Very) Brief Overview of Automation

Before we get started, we should take a moment to explain briefly how Automation works.

Automation uses **generators** - special rules that tell Structure what issues to show you from Jira and where to place them within your structure. Every time you open Structure, these generators will run again and completely rebuild your structure, based on the current information available. In this way, you know that your structure is always up-to-date and relevant. To learn more about how Automation works and the types of generators available, see Generators(see page 140).

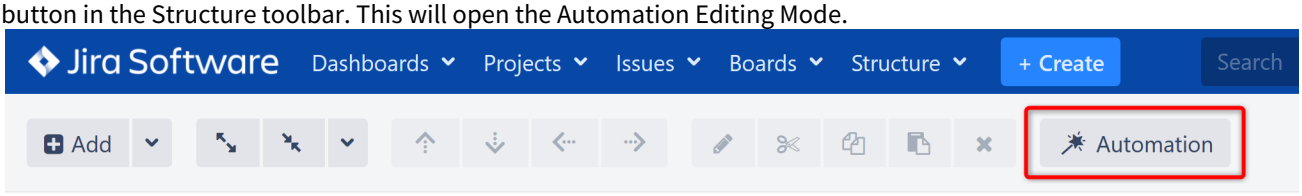
For now, we'd like to show you you how to build two of our most popular (and useful) automated structures, both of which can be built in just a few minutes.

- Top Down Automation to Manage Issues Across Multiple Projects(see page 59)
- Bottom Up Automation for Backlog Grooming(see page 64)

2.2.3.2 Top Down Automation to Manage Issues Across Multiple Projects

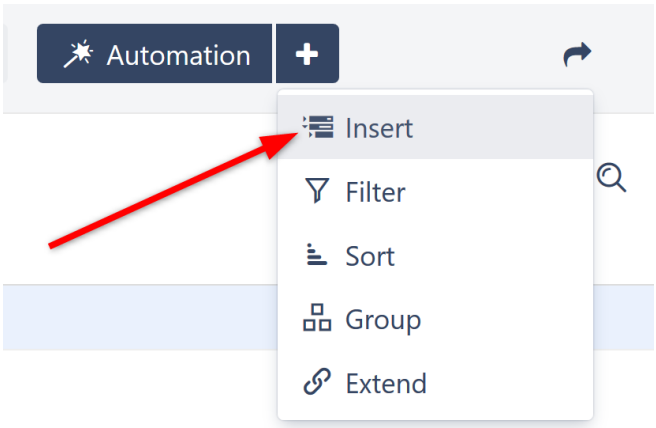
One of the great advantages to using Structure is the ability to manage issues from multiple projects in a single location. The following guide will walk you through one approach for adding Epics, Stories, and Tasks from multiple projects into a single structure.

Starting with a brand new, blank structure (see Creating Your First Structure(see page 57)), click the **Automation** button in the Structure toolbar. This will open the Automation Editing Mode.



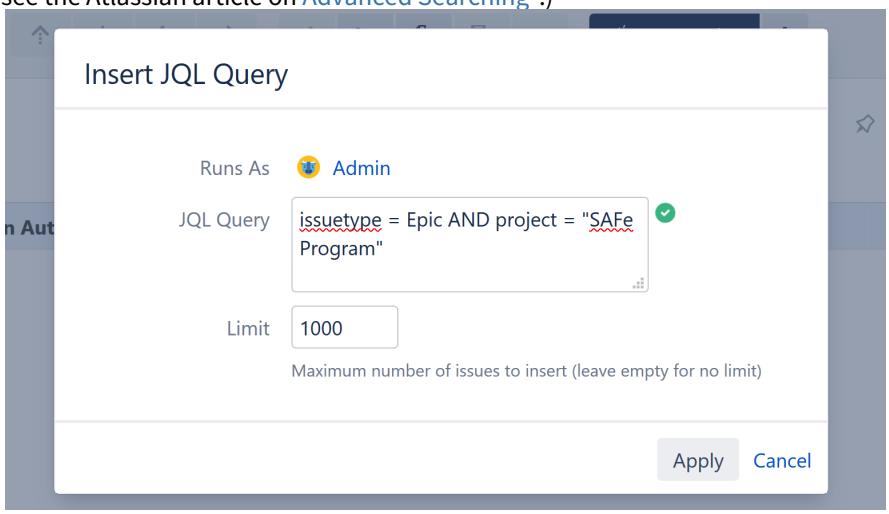
Step 1: Insert Epics

Once Automation Editing Mode is on, you'll see a new '**+**' icon next to the Automation button. Click this and select **Insert**.



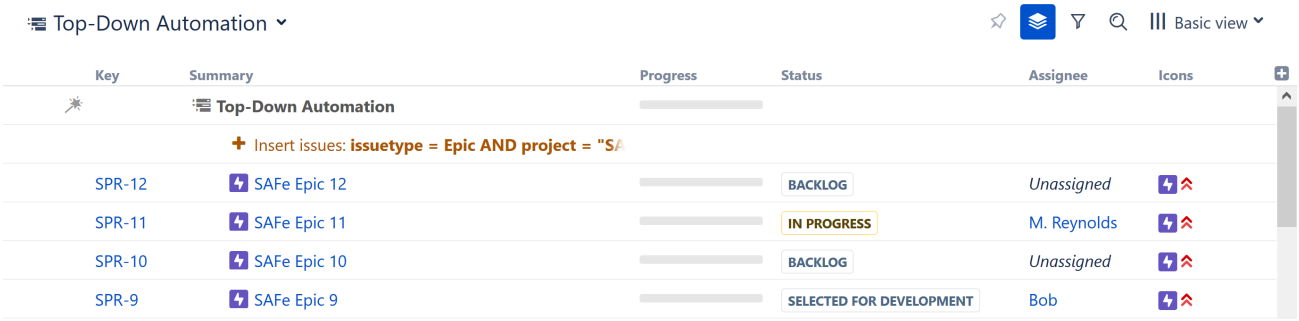
You can import items from an Agile board or existing structure, or you can use a JQL query or text search. Since we want to view issues from across multiple projects, let's use a simple JQL Query: "Issuetype = Epic". If you want to narrow your results to specific projects, you can specify that within your query as well. (To learn more about JQL,

see the Atlassian article on Advanced Searching[3].)



For large Jira instances, you may have hundreds of thousands of issues. While unlikely a problem for Epics, you may want to limit the number of issues to insert if you experience performance problems or simply want more manageable results. To do so, simply adjust the value in the **Limit** field.

Once you click **Apply**, the Insert generator will pull all issues that match your query into your new structure. In this case, they will add the Epics from our specified project.



While you are in Automation Editing Mode, all generators are listed in red.

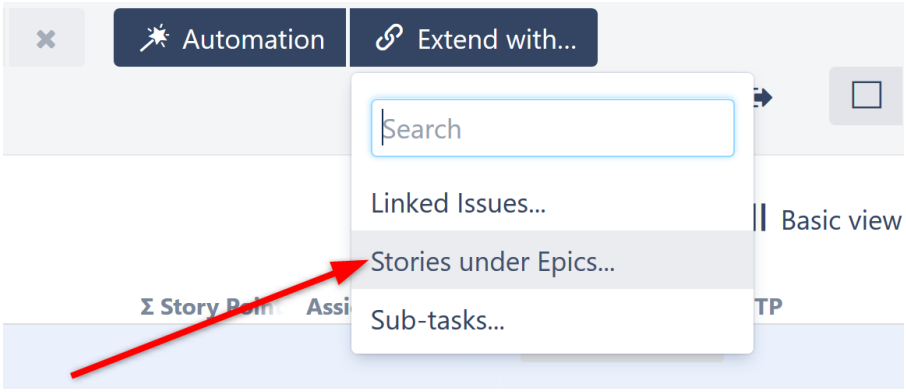
- To make changes to a generator, double-click it.
- To delete a generator, highlight its row and click the delete icon (**x**) or delete key.

Step 2: Extend with Stories

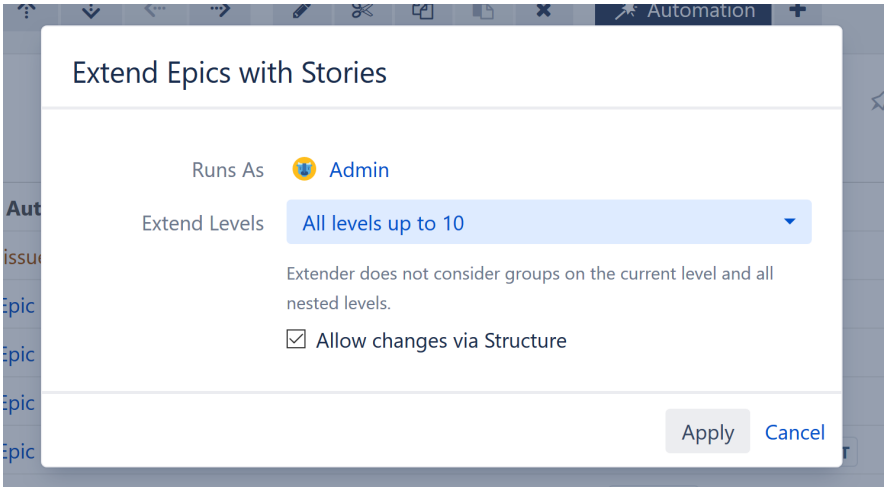
Now it's time to **Extend** our results to include Stories and Sub-tasks. To do this, make sure the top line of your structure, containing the structure's name, is still highlighted (this makes sure we are applying the generator to the entire structure) and click the Add Generator button **(+)** again. This time select **Extend**.

In the "Extend with..." menu, select **Stories under Epics**.

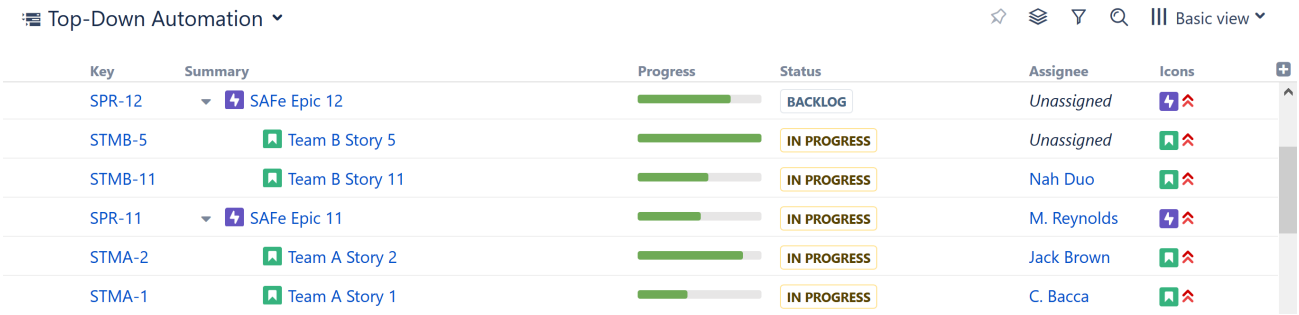
3 https://confluence.atlassian.com/jiracoreserver073/advanced-searching-861257209.html



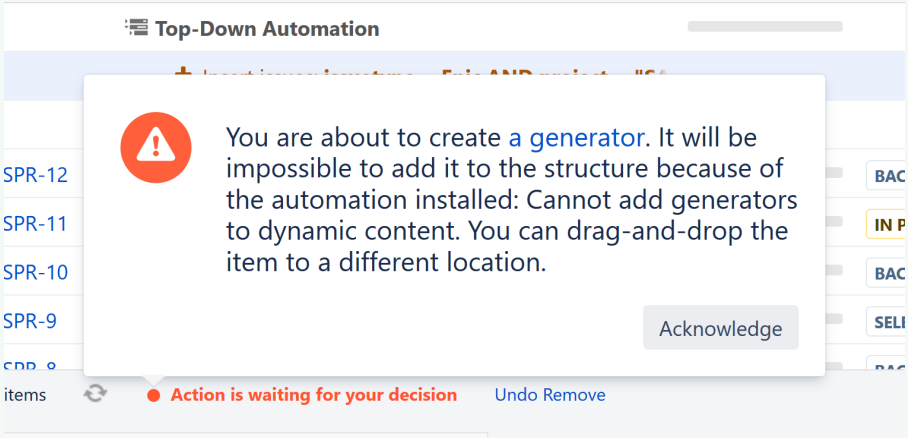
The Extend generator gives you the option to "Allow changes via Structure." When selected, moving a story from one epic to another within Structure will also move the story within Jira. This option is selected by default.



Click **Apply**. Some (maybe all) of your epics should now have a small arrow next to their Summary. Click the arrow to expand to the next level in the hierarchy – in this case, that's the Stories under each Epic.



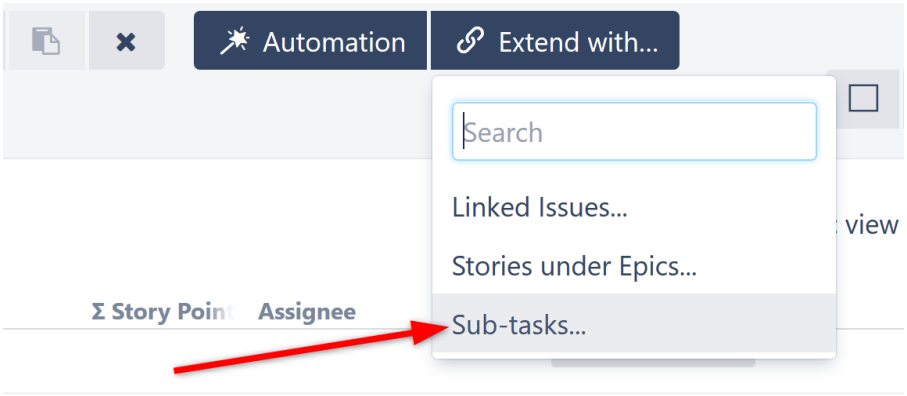
If you forget to highlight the top line of your structure before adding the Extend generator, you may get the following error:



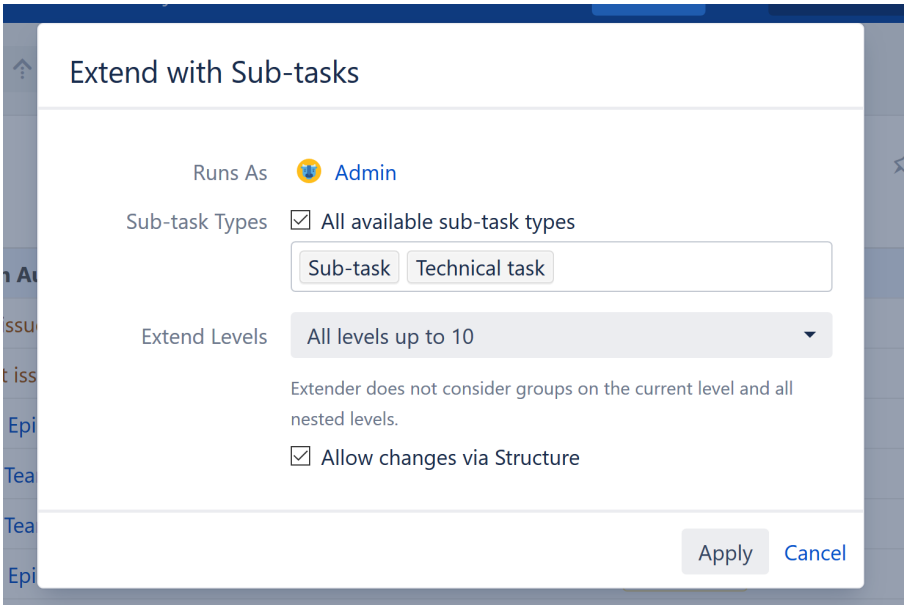
Since we started with a global generator (affecting the entire structure), all other generators need to be global as well, to avoid any conflicts. Simply highlight the top line of your structure (where it says the structure's name), and try again.

Step 3: Extend with Sub-tasks

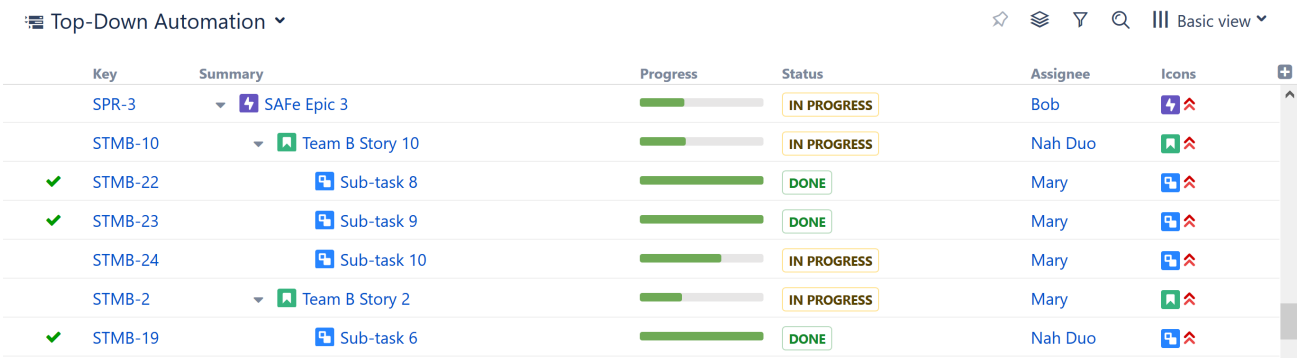
Finally, we'll add Sub-tasks under Issues. Once again, make sure the top line of your structure is still selected (highlighted), click the '**+**' next to the Automation button and select **Extend**. This time, select **Sub-tasks**.



You can choose to only add specific types of sub-tasks, or choose **All available sub-task types**. You can also select how many levels to include, as well as whether or not you want to allow changes within Structure. Once you've made your choices, click **Apply.**



Your issues should now appear in a hierarchy, based on the Automation rules we selected. If you want to re-arrange items or move a sub-task from one story to another (or copy it from one to another), you can do so from within your structure. (We'll cover that in Working with Structure(see page 68).)



You've built your first Automated structure! Now you can track tasks across your organization, easily follow the progress of stories or epics, and so much more.

As we mentioned at the start of this guide, this is just one of many ways to build a top-down structure. Depending on your needs, you may want to start with Themes or Initiatives instead of Epics, and work your way down from there. You may want to add Linked tasks. Or you may want to include other types of generators – we don't have space to cover all those options in this guide, but you can explore all your options in the Generators(see page 140) section of our Structure User's Guide.

> We mentioned a few times that before adding a new generator, you should make sure the top line of your structure is highlighted. This makes sure the generator applies to the entire structure. You can also add automation to specific locations in a structure – for instance, you could create a folder for a particular project and import all the epics from that project into there. In this case, you would highlight the folder before adding the automation.

Next Steps

Next, we'll take a look at how to create a bottom-up structure, which can be very useful for tracking projects and tasks across teams or departments.

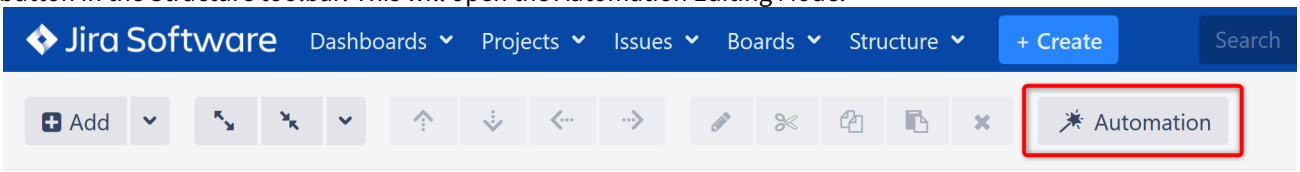
Bottom Up Automation for Backlog Grooming(see page 64)

2.2.3.3 Bottom Up Automation for Backlog Grooming

Now that you've seen how easy it is to track tasks across multiple projects using the Extend generator, let's look at how we can use the Group generator to look at work assigned to specific teams or team members – and quickly manage your backlog.

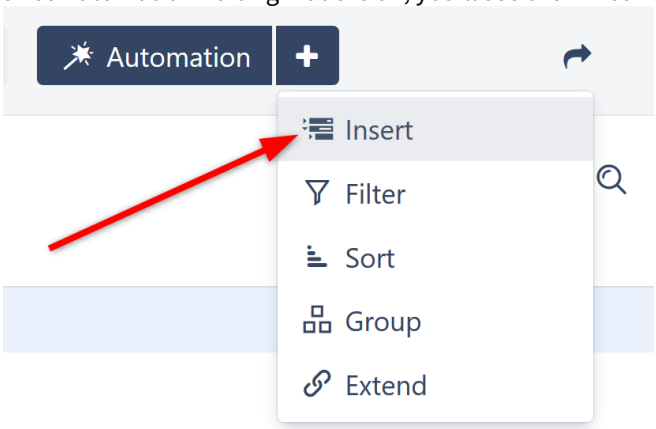
In the last tutorial, we started at the top of our hierarchy (Epics) and worked our way down. This time, we're going to start at the bottom and insert all of our active stories.

Starting with a brand new, blank structure (see Creating Your First Structure(see page 57)), click the **Automation** button in the Structure toolbar. This will open the Automation Editing Mode.



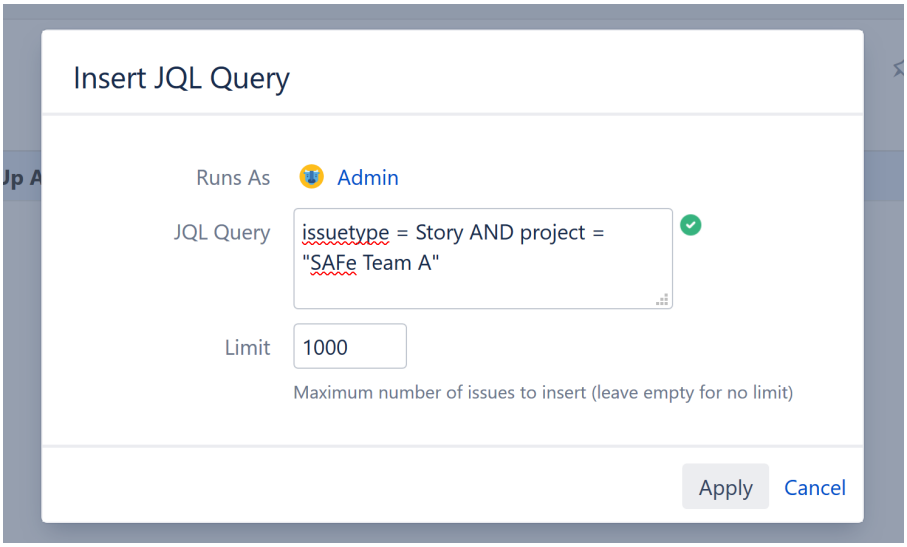
Step 1: Insert Stories

Once Automation Editing Mode is on, you'll see the '**+**' icon next to the Automation button. Click it and select **Insert**.

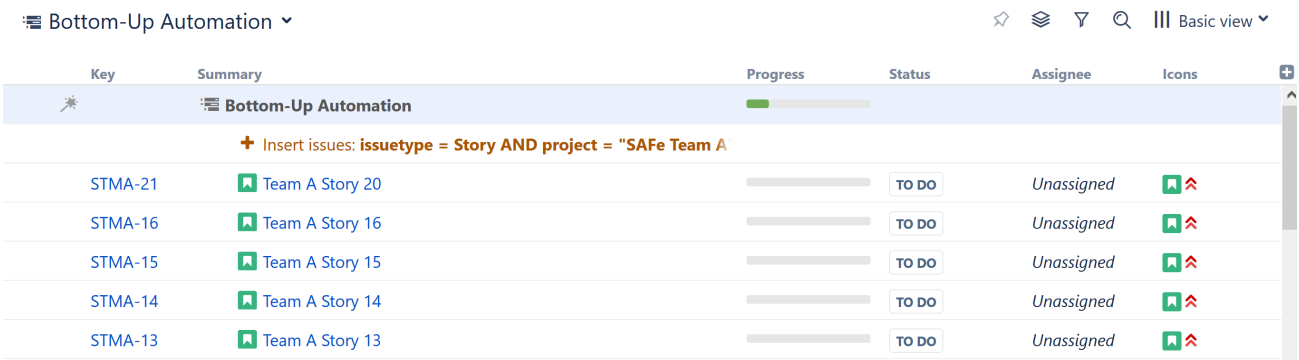


This time, we'll use the JQL Query: "Issuetype = Story". Just as with our top-down structure, you can also narrow your results by specifying projects (see Advanced Searching[4] for more information about using JQL).

4 https://confluence.atlassian.com/jiracoreserver073/advanced-searching-861257209.html



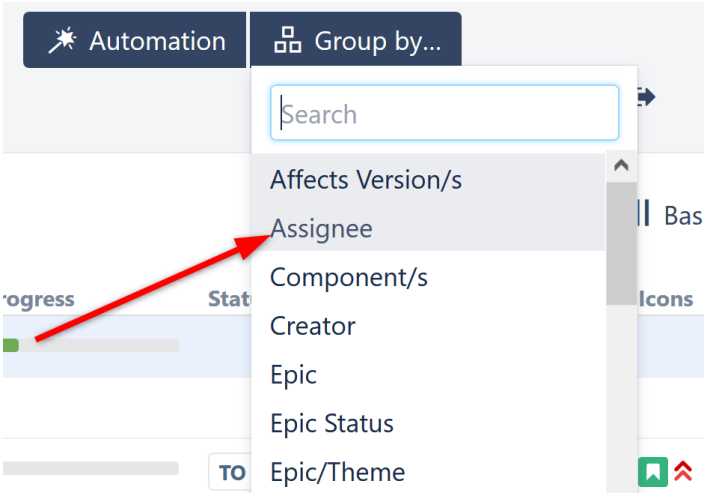
Once again, the Insert generator will pull all issues that match our query into the new structure. In this case, they will add the stories from our specified project.



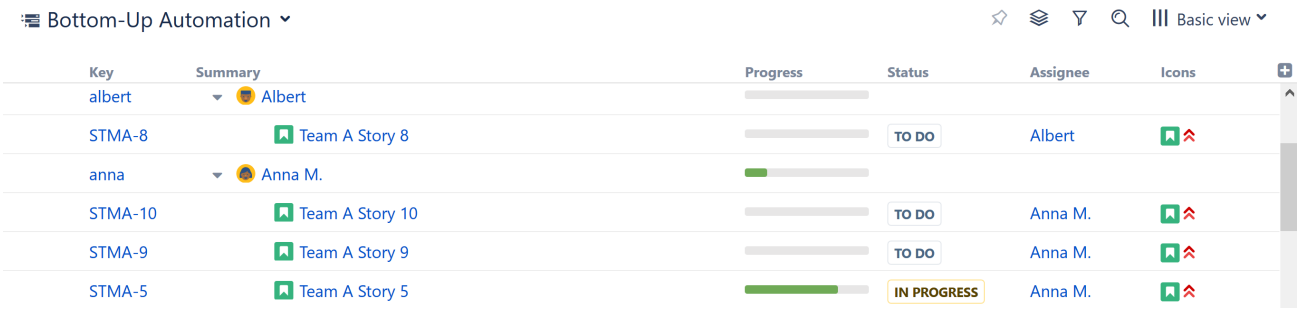
These will serve as the bottom-most level of our hierarchy, and we will use the Group generator to add levels above this.

Step 2: Group by Assignee

Make sure the top line of your structure is still highlighted and click the Add Generator button **(+)** again. This time, select **Group**. In the "Group by..." menu, you will see more options than were available for the Extend generator. You can group issues by nearly any attribute. But for the purpose of this guide, we're going to begin by grouping all of our stories by their assignee. So in the drop-down list, choose **Assignee**.



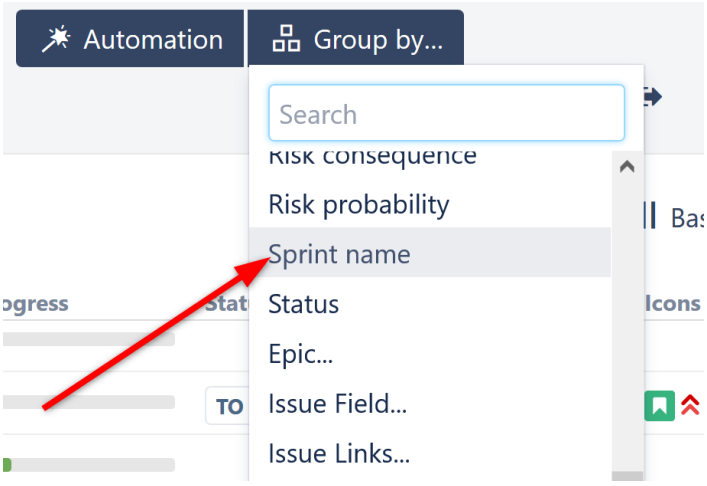
At this point, all of your stories should appear beneath their respective assignee.



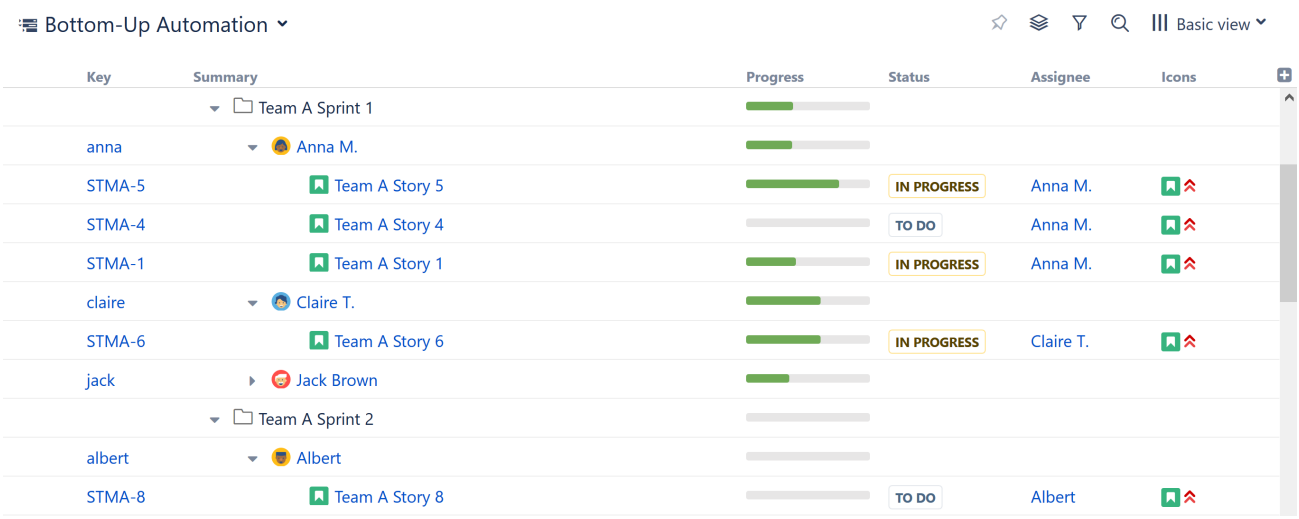
Now we'll add another level to our hierarchy, so we can view the work being done for each sprint.

Step 3: Group by Sprint

Make sure the top row of your structure is still highlighted and return to the Add Generator menu. Select **Group** again. But this time, let's group by **Sprint name**.



Once you apply this new generator, the top-level of your hierarchy will show all of your sprints. Under each of these, you can see which team members are assigned to which stories, and which stories are still unassigned.



Any stories that have yet to be assigned will be placed under an **Unassigned** folder. You can assign (or reassign) items simply by dragging them to a new location in the structure.

Taking it Further

At this point, you already have an incredibly useful structure. But you don't have to stop there.

As we noted above, you can group by nearly any attribute:

- Perhaps you want to see who worked on which projects for a fix version. Simply add another Group generator, this time grouping by Fix Version.
- Or you could group all of your results by Priority.

Take some time to explore the many options available under the Group extender. And remember, you don't have to choose just one configuration. You can create as many structures as you need. Set up one that just looks at assignees, and then have others that dig deeper in whatever way you need to see the information. Since generators are re-run every time you open the structure, you'll always have the up-to-date information you need, just the way you need it.

> ⚠ These are just examples of a few ways you can build structures using Automation. There are hundreds of other possibilities, allowing you to create the perfect hierarchy for your business needs – or several perfect hierarchies! In this guide, we focused on building down with extenders and building up with groupers, but you don't have to work in just one direction. You can start in the middle and add both extenders and groupers. If you really want to customize things, you can add multiple inserters, group different levels by different attributes, sort specific levels, and more.
>
> To learn more about some of these advanced steps, check out our full articles on Generators(see page 140).

Next Steps

If you're interested in learning about adding items to a structure manually, continue on to Building a Structure Manually[5]

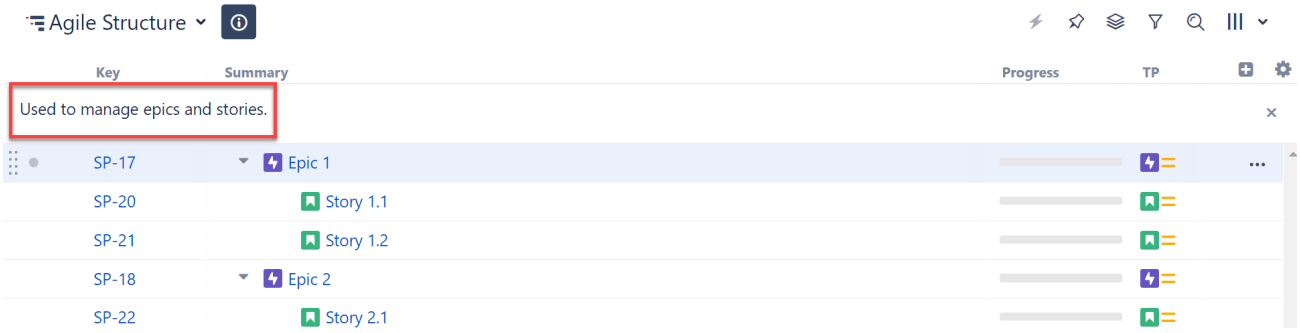
If you would rather keep things simple, jump ahead to Working with Structure(see page 68)

5 https://wiki.almworks.com/display/structure/.Building+a+Structure+Manually+v8.2

2.3 Working with Structure

2.3.1 Structure Description

The first time you open a structure, its description is shown at the top of the Structure panel, so you can quickly identify its purpose.



To close the description, or open it again, click the info button beside the structure's name

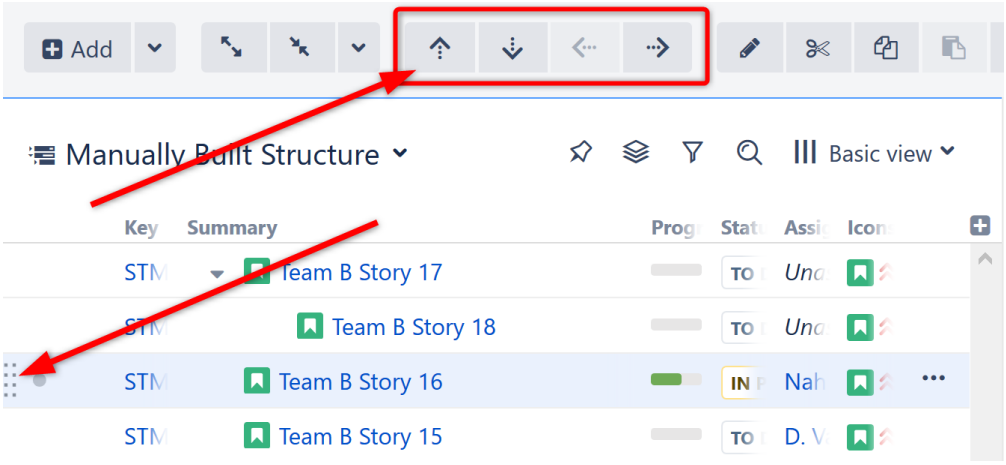
2.3.2 Moving, Adding and Removing Items from a Structure

Most actions within Structure can be done using the mouse, Structure Toolbar, or keyboard shortcuts.

2.3.2.1 Moving Items

To move an item, highlight it within your structure and:

- Use the arrow keys in the Structure toolbar,
- Use the drag bar to the left of the item to drag-and-drop it into a new position, or
- Use the keyboard: hold the **ctrl key** and press **up, down, left or right**.



Moving an issue to the **Right** places it lower in the hierarchy. Moving it to the **Left** places it higher.

2.3.2.2 Adding Items

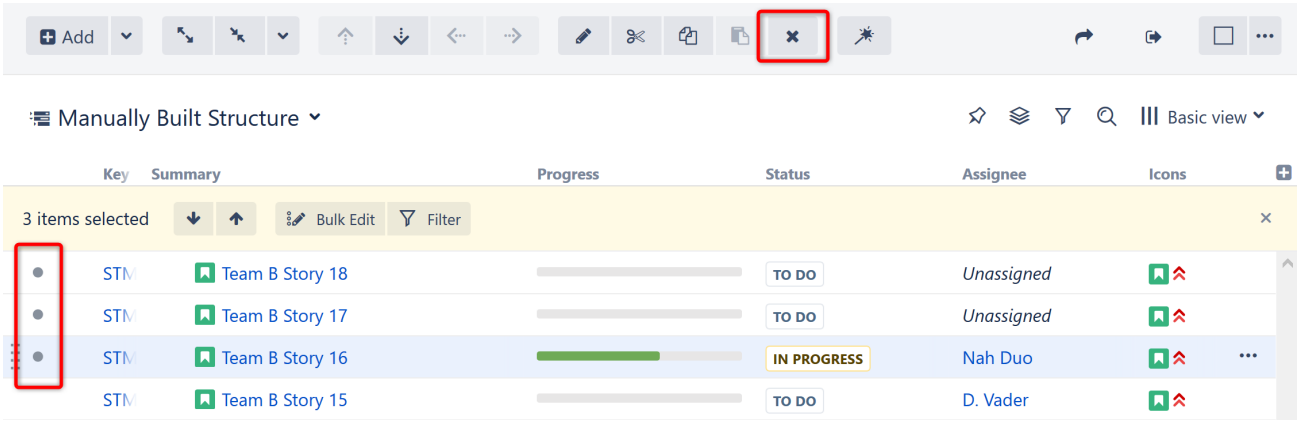
To add items (issues, folders, etc.) to a structure:

- Use the Add menu in the Structure Toolbar,
- Drag and drop the item from Jira or another structure (see Building a Structure Manually[6]), or
- Use Automation (see Building a Structure with Automation(see page 58)).

2.3.2.3 Deleting Items

To delete a single item from a structure, highlight the item and click the **X** (delete) button in the Structure Toolbar or use the **delete key** on your keyboard.

To delete multiple items, select each item by clicking the gray dot at the beginning of each item row (see Selecting Multiple Items(see page 113)). Once you have selected all the items you want to remove, click the delete key in the toolbar or use your keyboard.



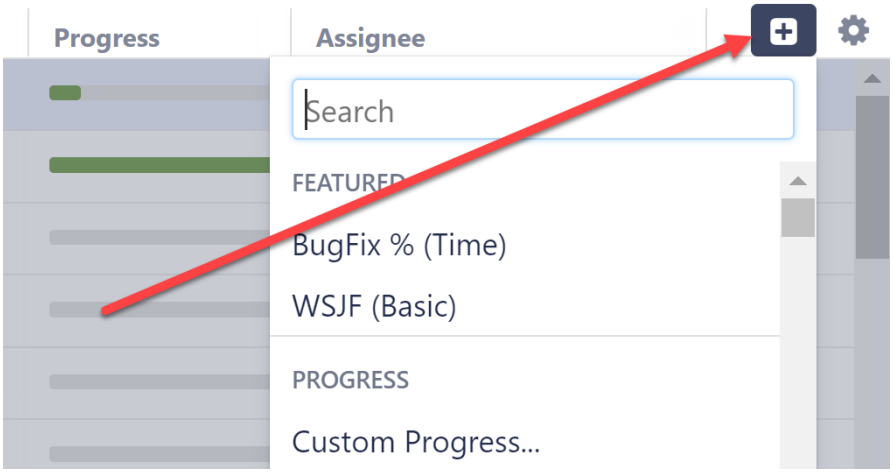
> Removing an issue from a structure does not delete the issue itself. It just removes it from the current structure.

2.3.3 Adding Columns

You can customize the Structure panel to include as many columns as you need. You can choose from a list of preset columns to display certain issue fields, attributes, formulas, and more. Or you can create your own custom columns.

To add a column to the Structure panel, click the **+** icon at the far-right of the column header.

6 https://wiki.almworks.com/display/structure/.Building+a+Structure+Manually+v8.2



To remove or customize a column, hover over its title in the column header and click the small triangle that appears.

2.3.3.1 Special Columns and Views

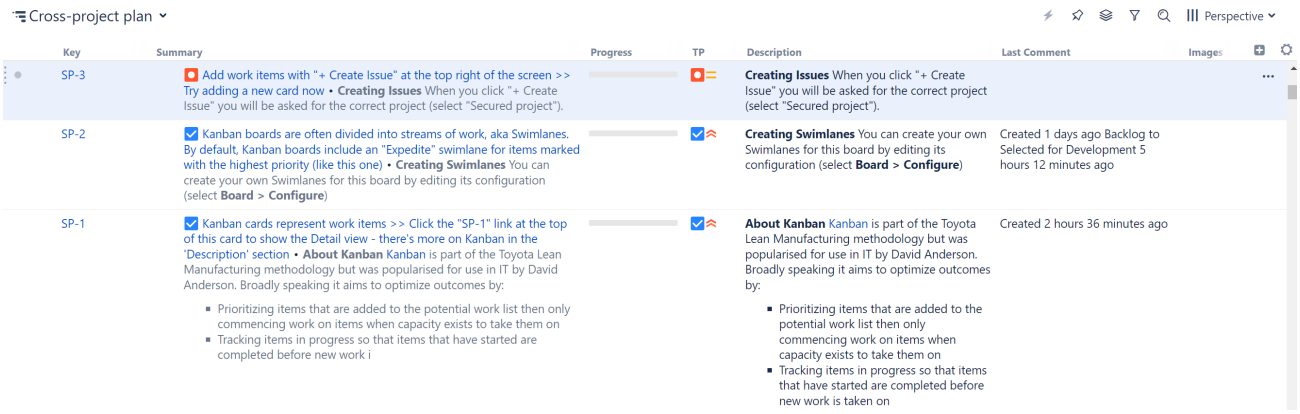
Structure also offers several very useful custom columns, including:

- Progress(see page 472) - Progress can be calculated based on the **Time Tracking**, **Status** or a custom field such as **Story Points**. To review (or change) your Progress calculation settings, hover over the Progress title in the column header and click the small triangle. You can find all the details on how the progress is calculated in the full documentation(see page 472).
- Work Logged(see page 499) - The Work Logged column displays the work logged for every issue over a selected period.
- Σ **Totals**(see page 461) - Structure offers a number of custom columns - including **Totals** columns. These special columns display the value for the selected field as a sum of the values of its children. This is especially useful for showing such things as **Estimates**, **Time Logged**, and **Story Points**.
- Formulas(see page 463) - Formula columns allow you to calculate values based on issue fields and other attributes using custom formulas. Structure offers several built-in formulas, or you can create your own.

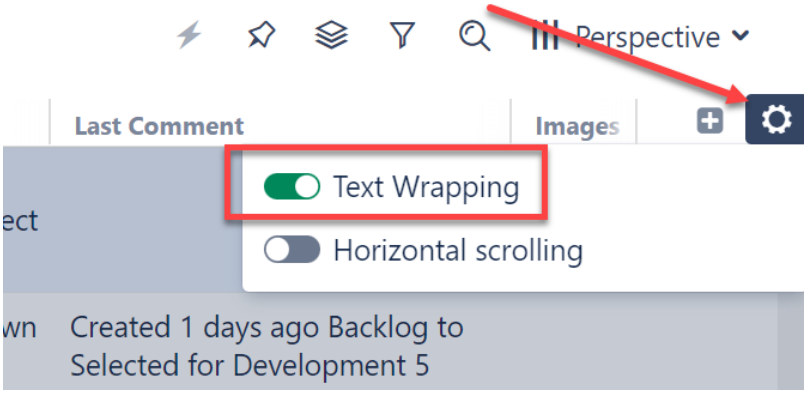
Once you've added all the columns you need, you can save them as a custom view. To do that, click the name of the currently selected view (in the right corner above the structure grid) and click the **Save As** link.

2.3.4 Text Wrapping

If you're working with fields that contain a lot of text or images, you can enable Text Wrapping to expand the height of each row to fit the content of those fields.



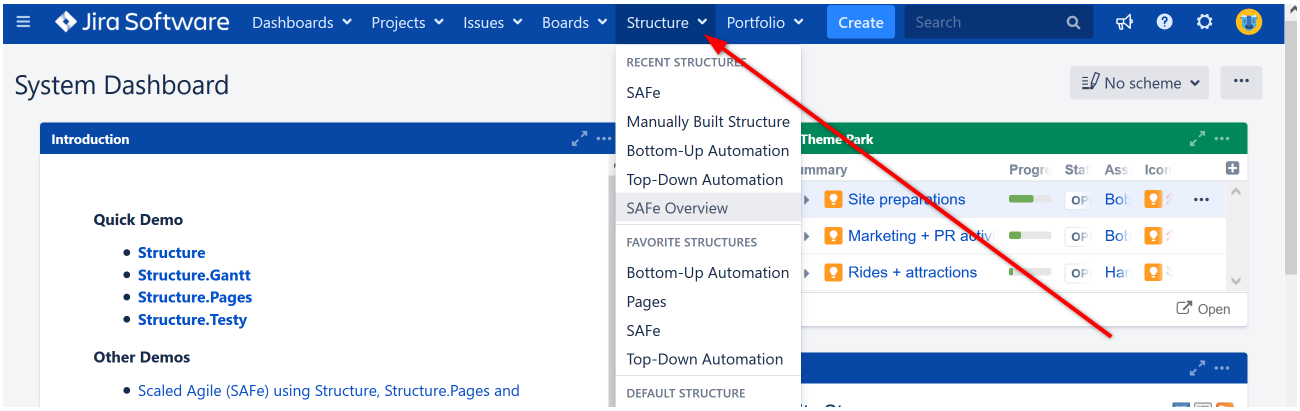
To enable or disable Text Wrapping, click the gear icon beside the Add Column button.



2.3.5 The Structure Board

In this tutorial, we have worked with Structure exclusively from the Structure Board. This is where you have the most freedom and capabilities, since your structures can work across any project you have permissions to view.

To open a structure in the Structure Board, select it from the Structure tab in the top menu.



If you don't see the structure you're looking for listed there, click **Manage Structures** at the bottom of the drop-down list and search for it on the Manage Structures page.

2.3.6 Structure in Other Locations

Structure is also available on:

- Issue Pages
- Project Pages
- Agile Boards
- Dashboard (as a gadget)

To learn more about where Structure can be found within Jira and the advantages and limitations of each location, see Jira Pages with Structure(see page 88).

2.3.7 Next Step

Next let's see how easy it is to work with individual issues without leaving your structure.

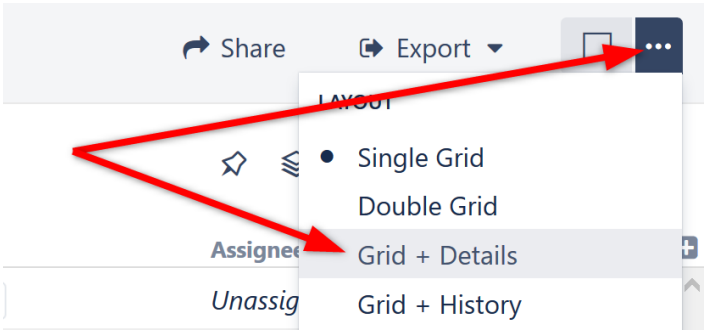
Working with Issues in Structure(see page 72)

2.3.8 Working with Issues in Structure

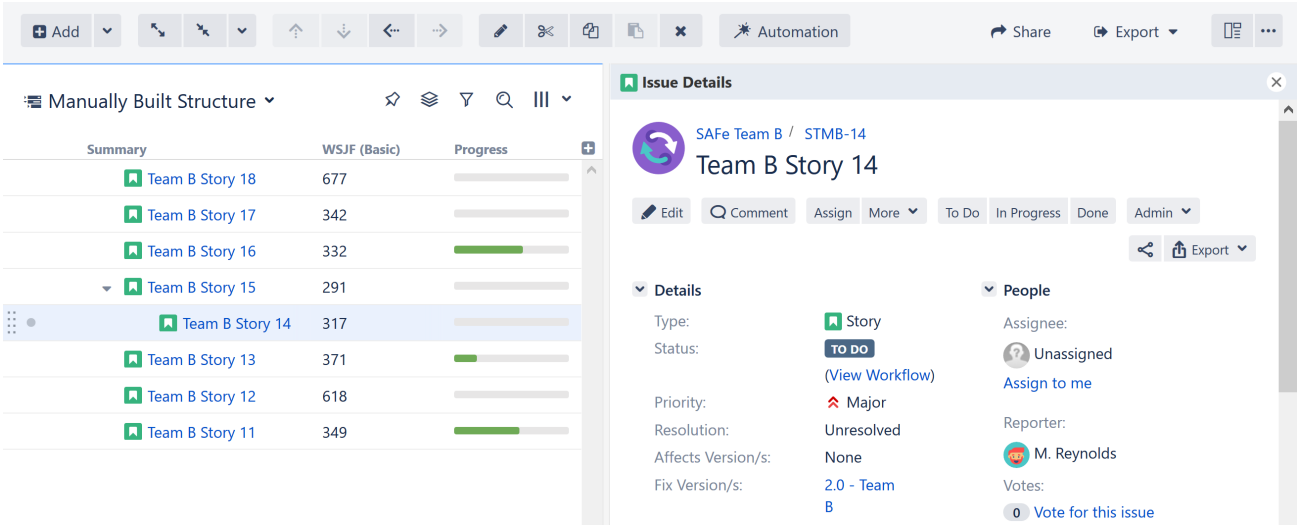
You can view issue details and make changes to issue properties directly from Structure.

2.3.8.1 Issue Details Panel

You can view full issue details within Structure, using the Issue Details Panel. To open the panel, use the Layout menu and select **Grid + Details**.



When you select an issue within the structure, its issue details will open in the right panel.

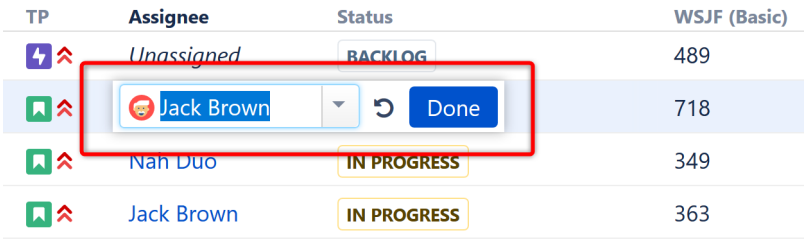


2.3.8.2 Editing Issues

There are multiple ways to edit issues directly from Structure.

Issue Columns

The simplest way to edit an issue is to make changes directly within your structure. If the field you want to change is visible from within the structure, simply double-click the current value to enter editing mode(see page 132).

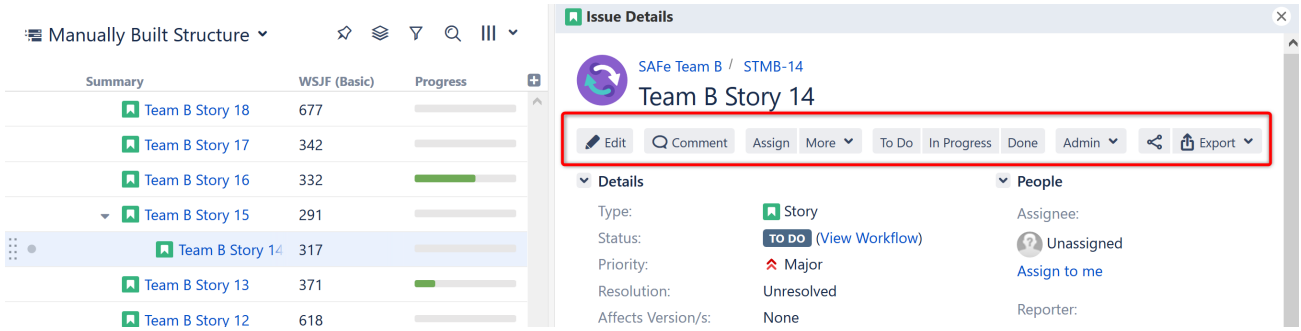


If the field is not visible, you can add the column(see page 501) by clicking the **+** icon at the top right of the Structure panel.

Issue Details Panel

In the details panel, you can work with the issue in the same way you can within the Jira Issue Navigator: edit[7], view and add comments[8], share[9], view history[10], view development information[11], and more.

7 https://confluence.atlassian.com/display/JIRA/Editing+an+Issue
8 https://confluence.atlassian.com/display/JIRA/Commenting+on+an+Issue
9 https://confluence.atlassian.com/display/JIRA/Emailing+an+Issue
10 https://confluence.atlassian.com/display/JIRA/Viewing+an+Issue%27s+Change+History
11 https://confluence.atlassian.com/display/JIRA/Viewing+the+Code+Development+Information+for+an+Issue

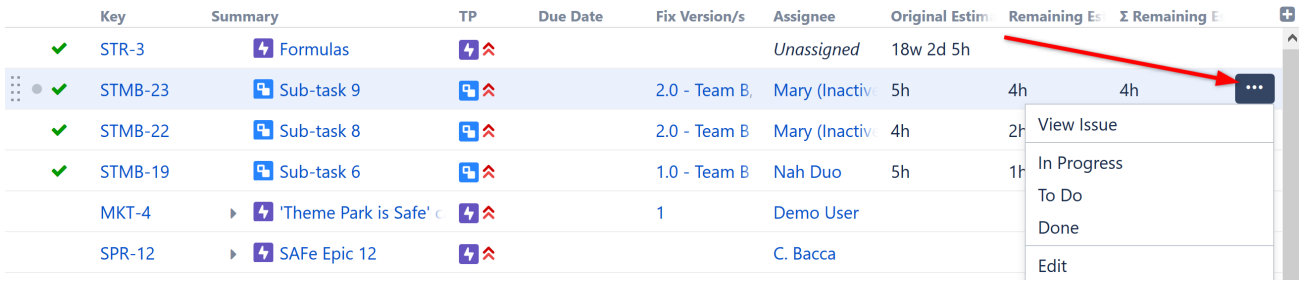


For specific information on working with and editing issues, please refer to the Jira documentation[12].

Jira Actions Column

Using the Jira Actions Column (far right column of the structure board), you can conduct many of the same actions available within the Issue Details Panel directly from you structure. This column works like the similar column on the Jira Issue Navigator page and lets you log work, apply workflow actions and use other Jira actions(see page 139) available for the issue.

As you hover over a row, click the **...** icon in the final column to view the Jira Actions menu.



2.3.8.3 Next Steps

Now that you know how to build structures and work with the items within a structure, it's time to see how easy it can be to search for issues and transform a structure to find just the information you need.

Search Filter and Sort(see page 74)

2.3.9 Search Filter and Sort

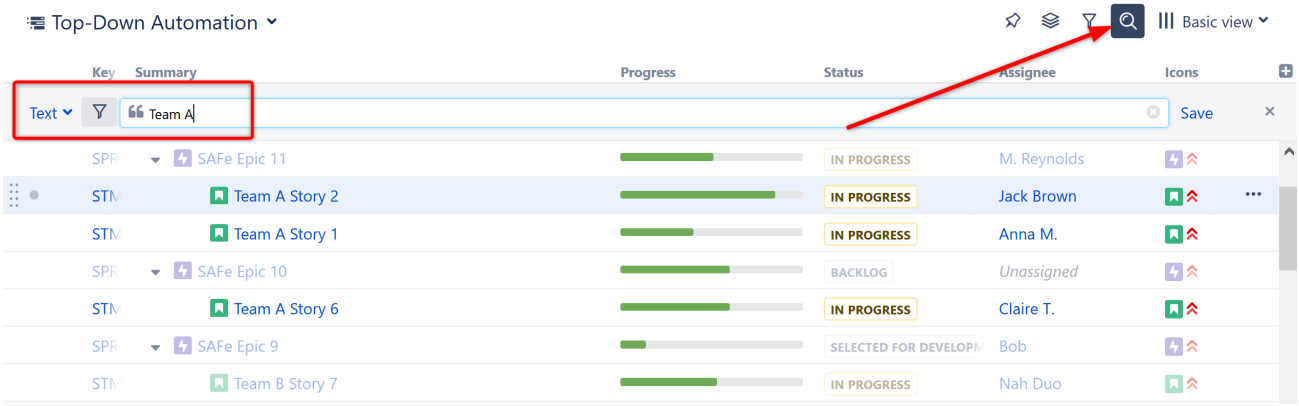
Once you add issues from several projects, you will likely want a way to organize, search and filter them, so you can focus in on specific information needs. Some of that can be handled using Generators(see page 140), which we introduced in Building a Structure with Automation(see page 58). But those operations affect the look and content of a structure for all users. What if you want to reorganize a structure temporarily? Or quickly hide issues you don't need to see at that precise moment?

This is where search and transformation come in.

12 https://confluence.atlassian.com/display/JIRA/Working+with+an+Issue

2.3.9.1 Search

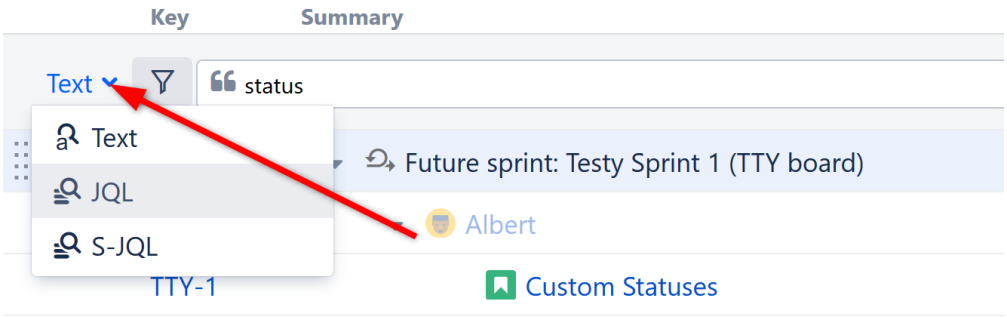
The Search function allows you to highlight issues within your structure. Click the **Search** button on the Structure Panel Toolbar and enter your query into the Search panel.



Notice that Search does not remove any issues from your structure, but instead grays out those issues that do not meet your query. This allows you to easily view your results within the context of the structure's hierarchy.

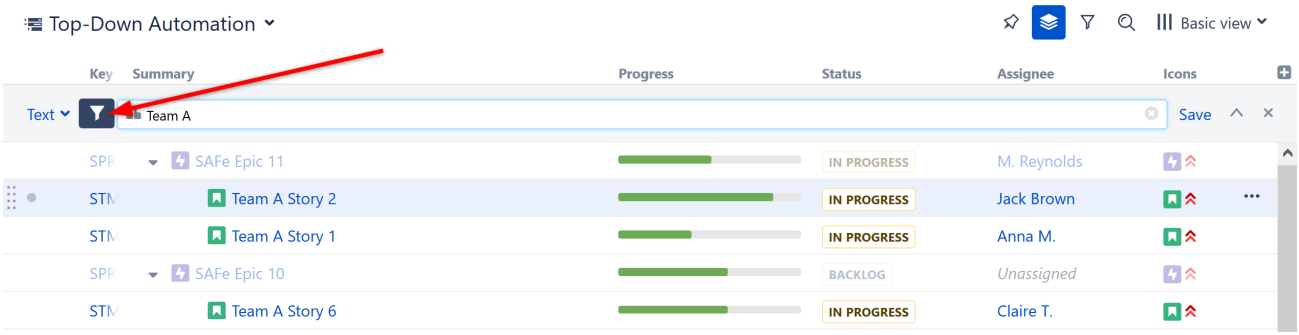
Search Options

You can search for issues(see page 205) in the current structure using a text search, JQL or Structure's own query language, S-JQL. To switch between these modes, click the name of the currently selected mode and select the one you need from the menu.



2.3.9.2 Filter

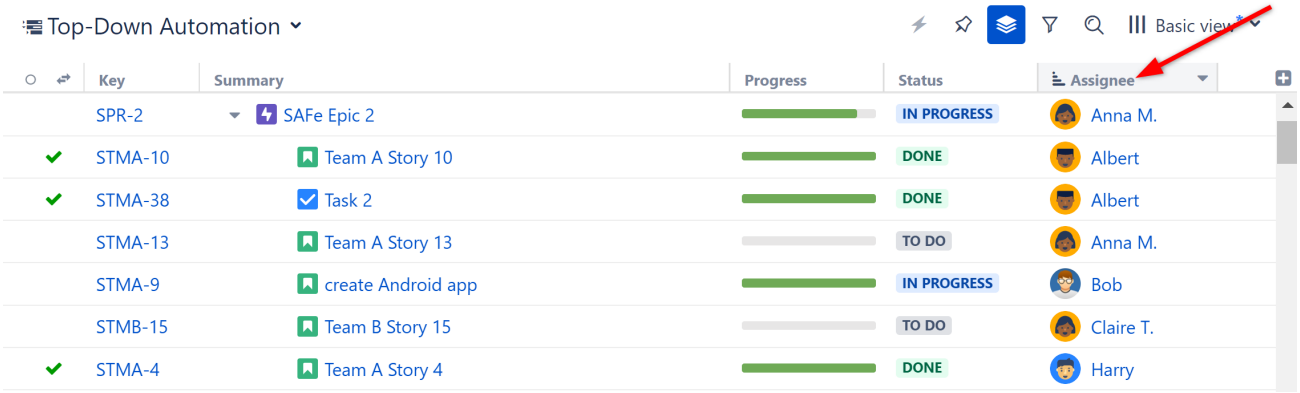
To hide issues from your results, click the Filter icon to the left of the query box. With the Filter activated, only your search results and their ancestors (issues above them within the hierarchy) will be displayed.



2.3.9.3 Sort

To quickly **sort** items in a structure, click the header of the column you want to sort by. Your structure will be sorted in ascending order, on every level.

- To sort in descending order, click the column header again.
- To remove the sorting, click the **Summary** column header.



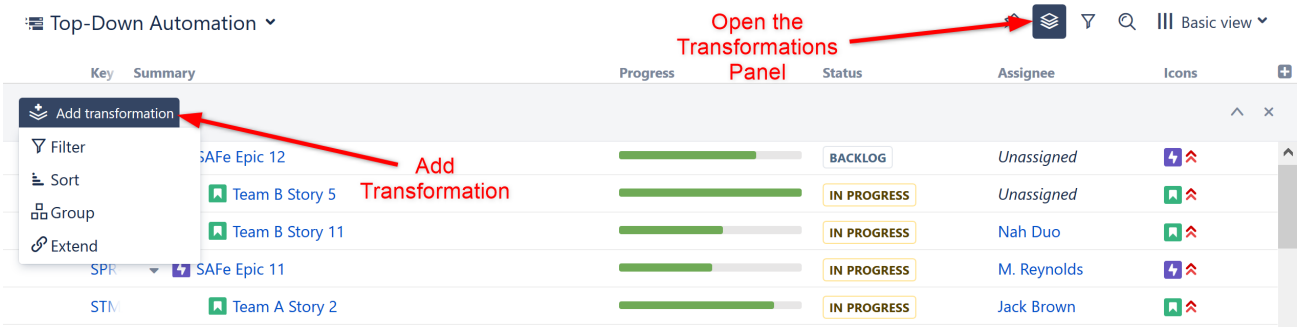
2.3.9.4 Transformations

You may have noticed that when you clicked the Filter button, the Transformation button  was also highlighted.

Transformations are actions that change the scope or order of your structure (such as hiding issues that fall outside your search scope). They allow you to do many of the same actions as Automation generators, but they are temporary actions that can easily be undone without any affect to your structure.

To apply a Transformation:

1. Click the Transformation icon
2. Click the Add Transformation button
3. Select the type of transformation you wish to apply and enter the appropriate parameters



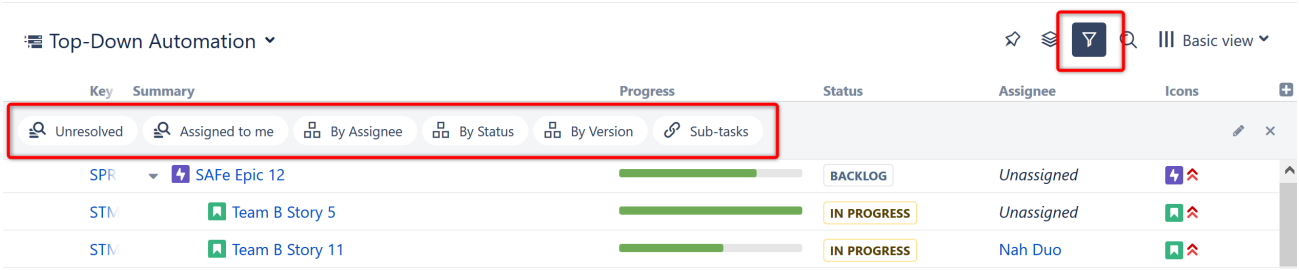
When you are finished with the transformation, simply click the **x** on the right side of the Transformations Panel and the transformation will be removed.

For more information, see Transformations(see page 215).

2.3.9.5 Quick Transformations

We chose a number of high-frequency transformations and added them to a Quick Transformations Panel. In just a couple of clicks, you can filter or organize your data to see only Unassigned issues, tasks assigned to you, and much more.

Simply click the Quick Transformations icon and selected the quick transformations you would like to apply.



You can change the quick transformations available on this list by clicking the edit button to the right of the Quick Transformations Panel.

To activate a quick transformation

1. Turn on Quick Transformations Panel
2. Click the desired transformation

To deactivate a quick transformation

- Click on the transformation again, or
- Close all quick transformations by clicking the "x" button or clicking the Quick Transformations icon again.

To learn more about Quick Transformations and how to create your own, see Quick Transformations(see page 220).

2.3.9.6 Next Steps

You have now mastered the basics of Structure and should feel confident in creating and managing customized structures for your organization. Next we'll quickly cover some of the other resources available to help you learn even more about Structure.

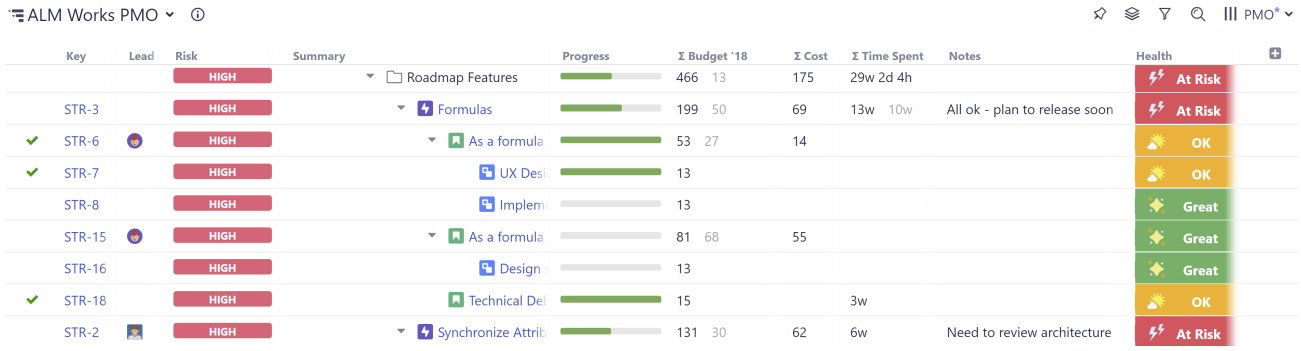
Help and Support(see page 80)

2.4 Getting the Most Out of Structure

Structure is more than just a way to organize your issues - it's a full-featured project management tool that allows you to view, track, and manage your data in highly customizable ways. Building your structure is just the first step - now it's time to add columns, formulas, and other features to get the most out of your data.

2.4.1 Add Custom Columns

Structure allows you to display tons of information about each issue - building your own custom reports.



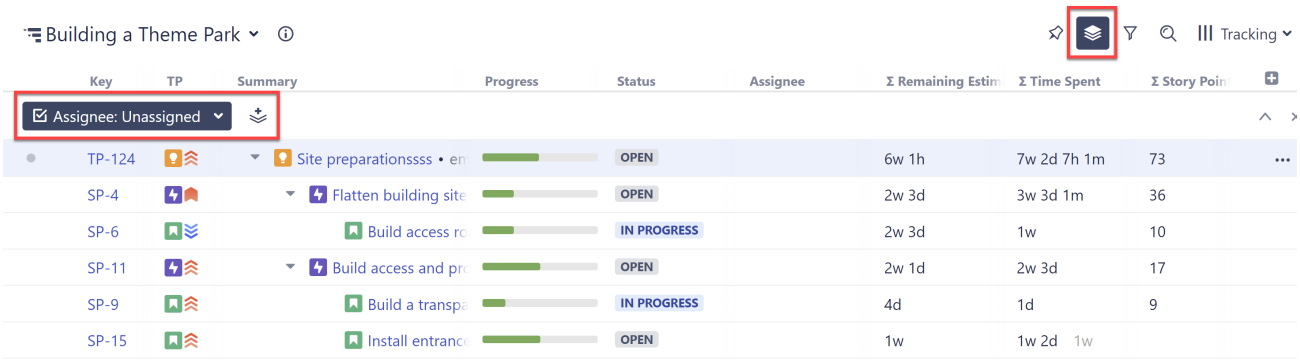
You can add a column for any Jira field, as well as many custom columns available only in Structure. Here are some of our most popular columns:

- Field columns(see page 461) - display and (optionally) aggregate the values in Jira fields
- Progress Column(see page 472) - calculate issue progress based on the values from the issue and its sub-issues
- Time in Status Column(see page 493) - calculate how much time issues spend in a particular status
- Notes Column(see page 470) - add additional information about an issue or project, without affecting anything in Jira
- Aggregate (Totals) Columns(see page 497) - aggregate values from numeric and time-tracking fields
- Formula Columns(see page 228) - perform simple or complex calculations based on issue fields, or compare values from multiple fields

Learn more: Adding Columns(see page 458)

2.4.2 Apply Transformations

Transformations allow you to reorganize your data to focus on specific issues or look at things from a different angle. Transformations are only temporary, and can only be seen by you - even if someone else is viewing the same structure.



The following types of Transformations are available:

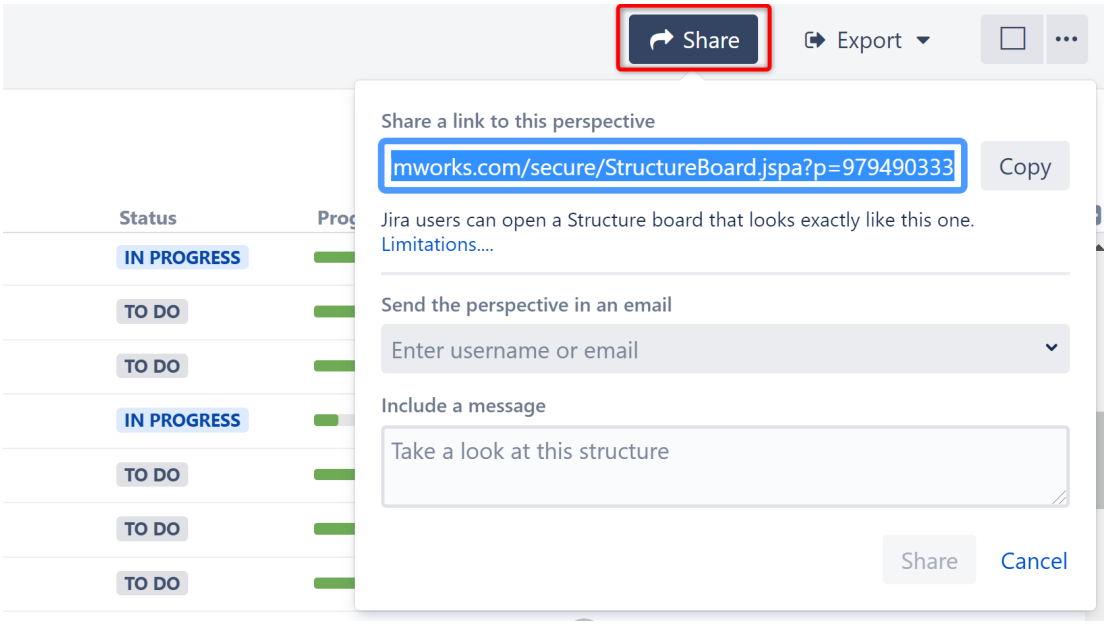
- Filter - filter the structure to focus in on specific issues
- Sort - reorder the structure based on the value of a field or Jira column
- Group - group issues by Jira fields or issue attributes
- Extend - add issues related to those issues already in the structure

Learn more: Transformations(see page 215)

Not ready to build your own? Use one of our predefined Quick Transformations(see page 220).

2.4.3 Share Your Structure

Now that you've created and customized your structure, share it with your team members, so they can see the same information you're seeing.



When you share a link to the structure, others will see the same hierarchy, issues, and columns as you - provided they have the necessary permissions.

Learn more: Sharing a Perspective(see page 97)

2.4.4 Try Structure Extensions

Structure extensions expand the power of Structure by providing a number of additional project management tools:

- Structure.Gantt[13] - Gantt charts and roadmaps for Jira
- Structure.Deliver[14] - Agile project forecasting
- Structure.Pages[15] - View and edit Confluence pages in Structure
- Structure.Testy[16] - Add lightweight testing checklists to Structure

View Documentation

- Structure.Gantt[17]
- Structure.Deliver[18]
- Structure.Pages[19]
- Structure.Testy[20]

2.5 Help and Support

You have now seen how easy it is to create and work with structures – and hopefully you're excited to try creating some new structures all on your own. Good luck and have fun!

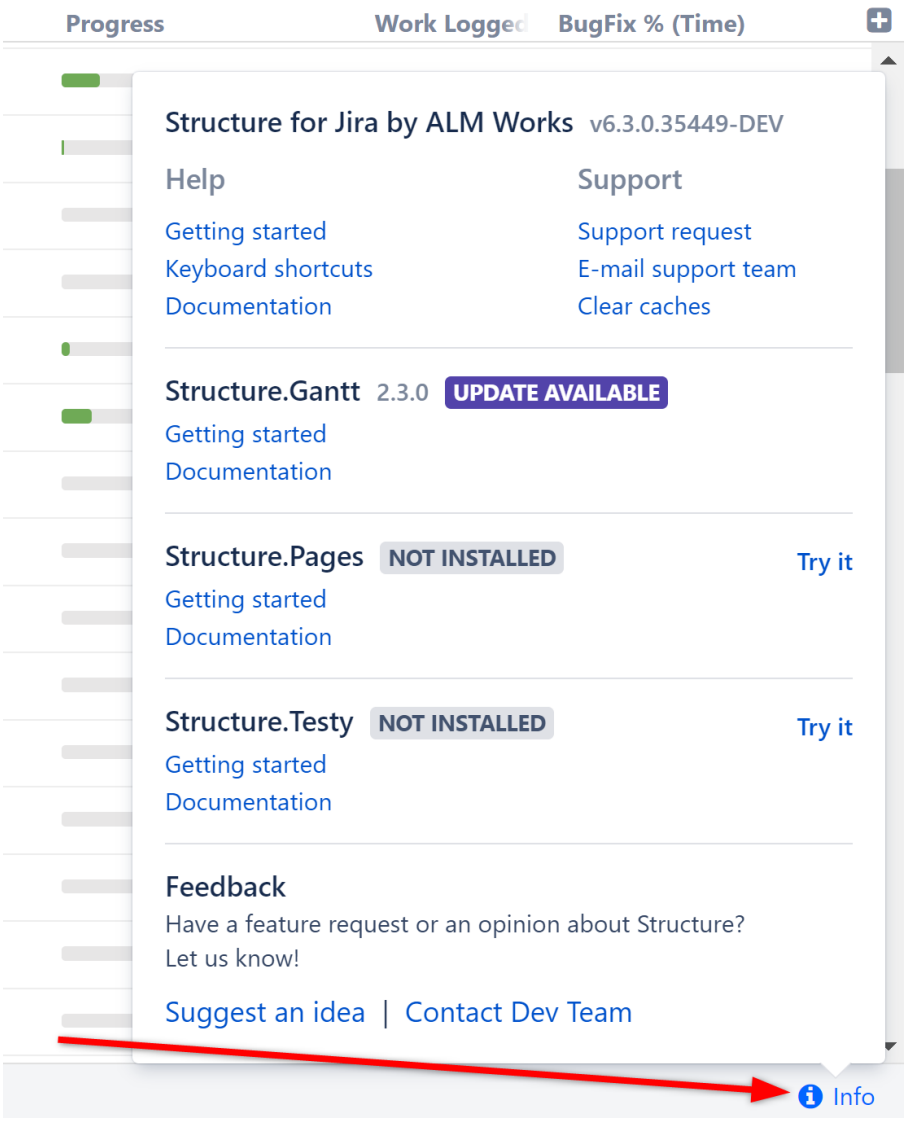
2.5.1 Available Resources

Should you have any questions or need assistance (whether regarding a feature of Structure or a personal need), we encourage you to refer to our Structure User's Guide(see page 82) or contact our support team.

2.5.2 Resources Within Structure

If you have a question or need assistance while working with Structure, click the **Info** link at the bottom right corner of the Structure Widget. This contains links to documentation and keyboard shortcuts, version and update information for Structure and Structure extensions, and links for our support services.

13 https://marketplace.atlassian.com/apps/1217809/structure-gantt-gantt-charts-roadmaps-for-jira?hosting=datacenter
14 https://marketplace.atlassian.com/apps/1224682/structure-deliver-agile-project-forecasting?hosting=datacenter
15 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=datacenter
16 https://marketplace.atlassian.com/apps/1212033/structure-testy-test-checklists?hosting=datacenter
17 https://wiki.almworks.com/documentation/gantt/latest/data-center-and-server
18 https://wiki.almworks.com/documentation/deliver
19 https://wiki.almworks.com/documentation/pages/latest
20 https://wiki.almworks.com/documentation/strtesty


Progress
Work Logged
BugFix % (Time)
Structure for Jira by ALM Works v6.3.0.35449-DEV
Help
Getting started
Keyboard shortcuts
Documentation
Support
Support request
E-mail support team
Clear caches
Structure.Gantt 2.3.0 UPDATE AVAILABLE
Getting started
Documentation
Structure.Pages NOT INSTALLED
Try it
Getting started
Documentation
Structure.Testy NOT INSTALLED
Try it
Getting started
Documentation
Feedback
Have a feature request or an opinion about Structure?
Let us know!
Suggest an idea | Contact Dev Team
Info

3 Structure User's Guide

If you're new to Structure, we suggest you begin by reviewing our Getting Started Guide(see page 56).

To learn more about specific features and functionalities, search for them here or browse our full list of articles:

3.1 Navigating Structure

3.1.1 Basic Concepts

We recommend you to get acquainted with a few important concepts to help shorten the learning curve.

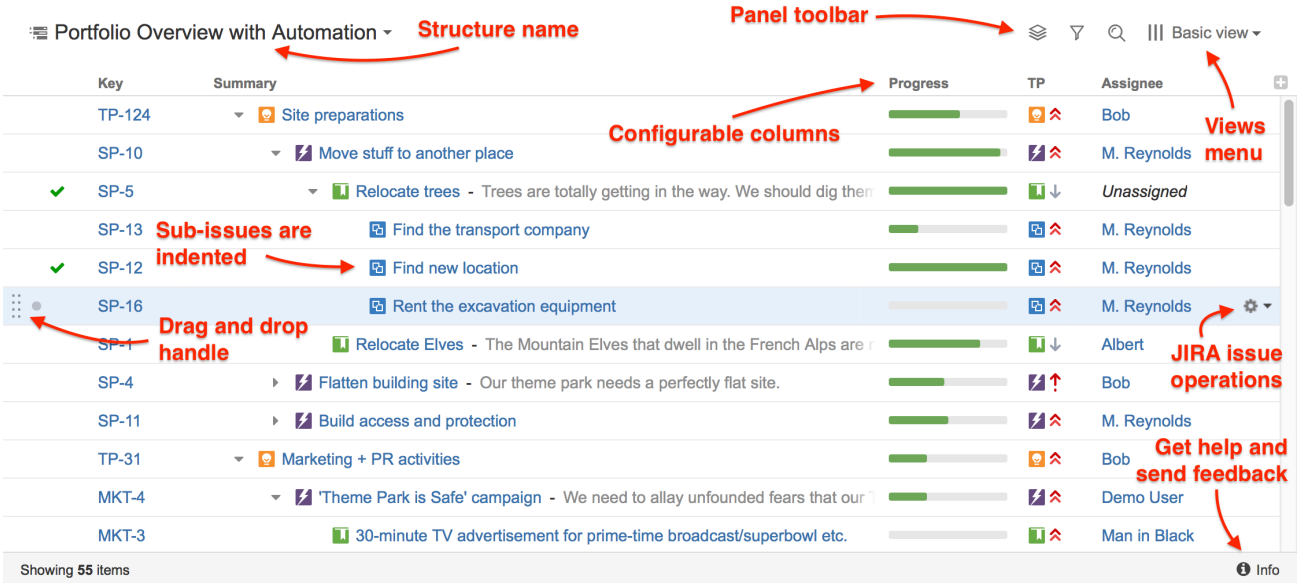
Structure (vs. structure)	*Structure* is the name of our product. In our documentation we differentiate between ***Structure***, the app, and the ***structures*** that you build with it using capitalization. When you see "*Structure*" with a capital "*S*" we are referring to the app. When you see "*structure*" with a lowercase "*s*" we are referring to the the structures you create in the app.

structures contain Jira issues	Think of a structure as a container that may be filled with Jira ***issues*** from a single, or multiple Jira projects. Within this container you may organize the issues into arbitrary groups of hierarchical lists. For example, you may wish to group your issues by type, by assignee, or by priority—or by some combination thereof. In fact, you may organize the issues in your structure any way you'd like.
items	Typically, Jira projects contain issues of many different types. For example, “bugs”, “tasks” or “activities”. Structure adds a few new, helpful, project management elements such as folders, memos and generators. Collectively, we refer to all of these (i.e., everything that appears in a structure) as ***items***.
automation / generators	As mentioned above, Jira issues may appear in a structure automatically. Our powerful ***Automation*** feature uses ***generators*** that automatically add Jira issues to a structure and organize them using issue attributes and business rules that you specify. Generators are also updated automatically, as issues in Jira change.
a view	We refer to a particular configuration of the columns that you decide to display in the Structure panel as a ***view***.
sub-items (and parent items)	When you place one item under another item in a structure it becomes a ***sub-item*** of the item above it. The item above the sub-item is the ***parent*** item. Sub-items may contain sub-items of their own, and those sub-items may contain still more sub-items, and so on. You may create as many levels of parent item / sub-item relationships as you wish in a structure. Importantly, these parent item / sub-item relationships may be different in different structures. The relationships may be created arbitrarily to suite your needs within a particular structure.
children	Sometimes we refer to sub-items as ***children*** (of the parent item).
Jira sub-tasks	Jira ***sub-tasks*** and Structure *sub-items* are conceptually similar, but they are not the same. Jira sub-tasks are a special type of Jira issue that includes a parent/child relationship within Jira. It may be desirable for sub-tasks to appear in your structures as sub-items of the relevant (parent) Jira task. However, this is not a requirement. There are no restrictions on Structure parent/child relationships, so sub-tasks may be placed anywhere in a structure, like any other other Jira issue type.

structures within structures	With Structure, ***you may add structures to other structures*** — i.e., a structure can be an item (see above) in another structure.

3.1.2 Structure Widget Overview

Structure displays issues as a hierarchical list. You can view as much or a little information about each item, by adding or removing columns(see page 501) to the Structure panel (or by selecting one of the predefined views).



Within the Structure widget, you can:

- Rearrange issues and adjust their position within the hierarchy
- Edit issues
- Perform Jira issue operations
- Search and filter issues
- Switch to a different structure(see page 85)

To learn more about performing each of these actions, see Moving Items within Structure(see page 108) and Working with Issues(see page 125).

> Structure widget is displayed on the Structure Board(see page 88), Issue Page(see page 90) and in other places in JIRA(see page 88).

3.1.2.1 Navigating Between Items

Navigating with Mouse

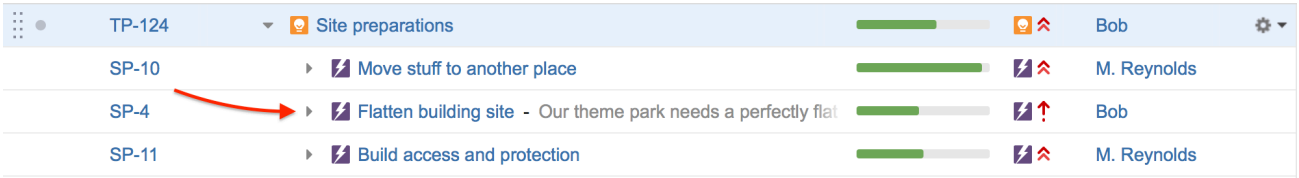
Using your mouse, you can:

- Select an item by clicking anywhere in the item's row, except on a link
- Scroll up and down
- View the Issue Details(see page 126) panel by clicking the Key or Summary link (to customize this behavior, see Viewing Issue Details(see page 126))

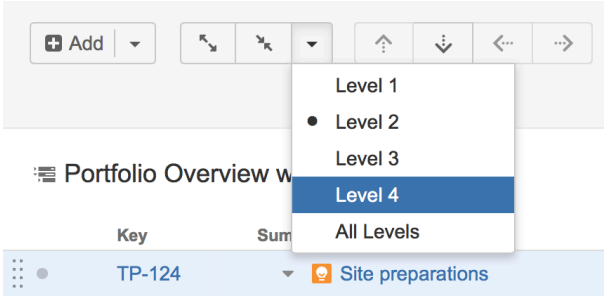
- Show or hide sub-items

Show/Hide Sub-Items

To show or hide sub-items, click the parent-item's **Expander** button, next to the item summary.



To expand or collapse the whole hierarchy, use the **Expand All** or **Collapse All** buttons in the toolbar. You can also expand the structure to a certain level by clicking the drop-down menu next to these buttons and selecting the desired level of depth.



For large structures, some items may be loaded on-demand to improve performance. As you scroll down or expand sub-items lists, you may experience a slight delay as the new data is loaded.

Navigating with Keyboard

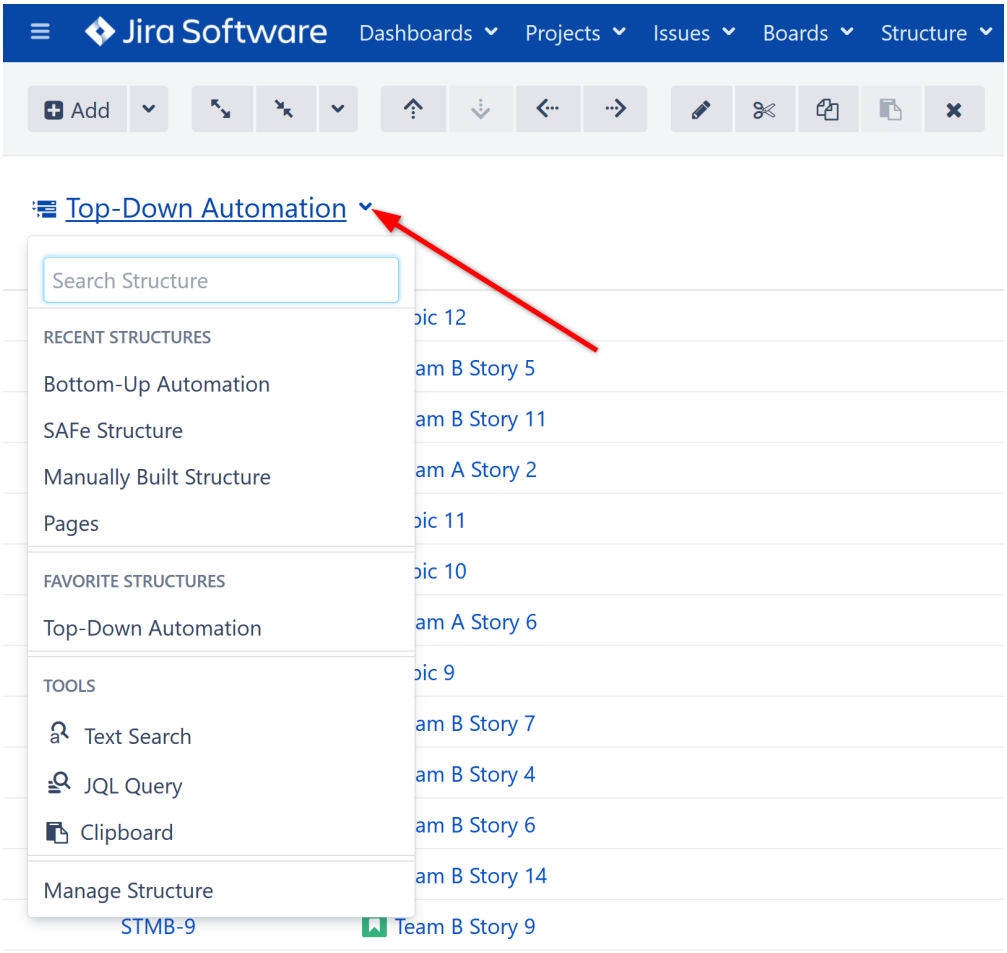
Using the keyboard, you can:

- Focus on the next or previous item in the list, using the **up or down arrows**
- Expand or collapse sub-items, using the **left and right arrows**
- Expand all sub-items, by pressing the **Plus** key twice
- Collapse all sub-items, by pressing the **Minus** key twice
- Move the selected item up or down in the hierarchy, or indent/outdent the item, using **Ctrl+Arrow Key**
- Open the Jira actions menu for the selected issue, by pressing **Alt+Down**

There are dozens of keyboard shortcuts(see page 116) available to simplify your Structure experience. Press **Ctrl+?** to view the shortcuts cheat sheet, or click **Info** at the bottom of the structure widget.

3.1.2.2 Switching Between Structures

If you need to switch structures without leaving the current Structure widget, instead of using the Structure menu(see page 99), click the name of the structure inside the widget:



If the structure you need is listed under the Recent Structures or Favorite Structures list, click the name to open it.

If the structure you are looking for is not listed, start typing the name into the **Search Structure** box at the top to see a list of matching structures.

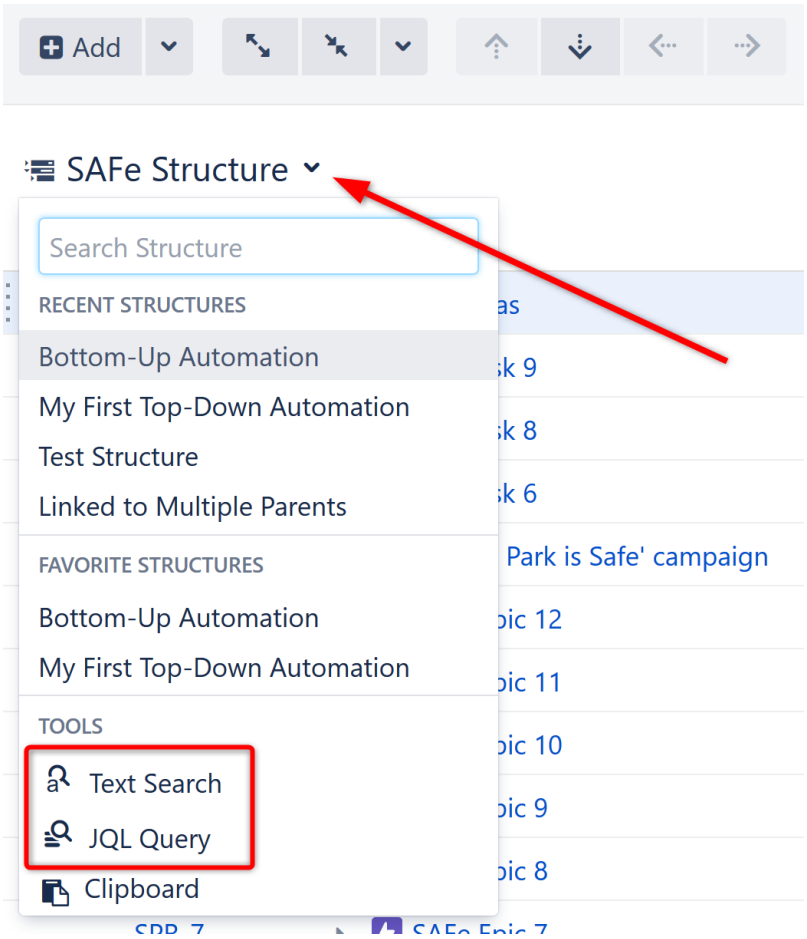
> Apart from choosing structures, you can also use this drop-down to run text and JQL searches, see the clipboard contents and - if you have Structure.Pages installed - search for Confluence pages. All of these options are available under the Tools section of the menu, when using Structure Board(see page 88) or Structure on the Project Page(see page 95)

3.1.2.3 Using Structure Widget for Searching

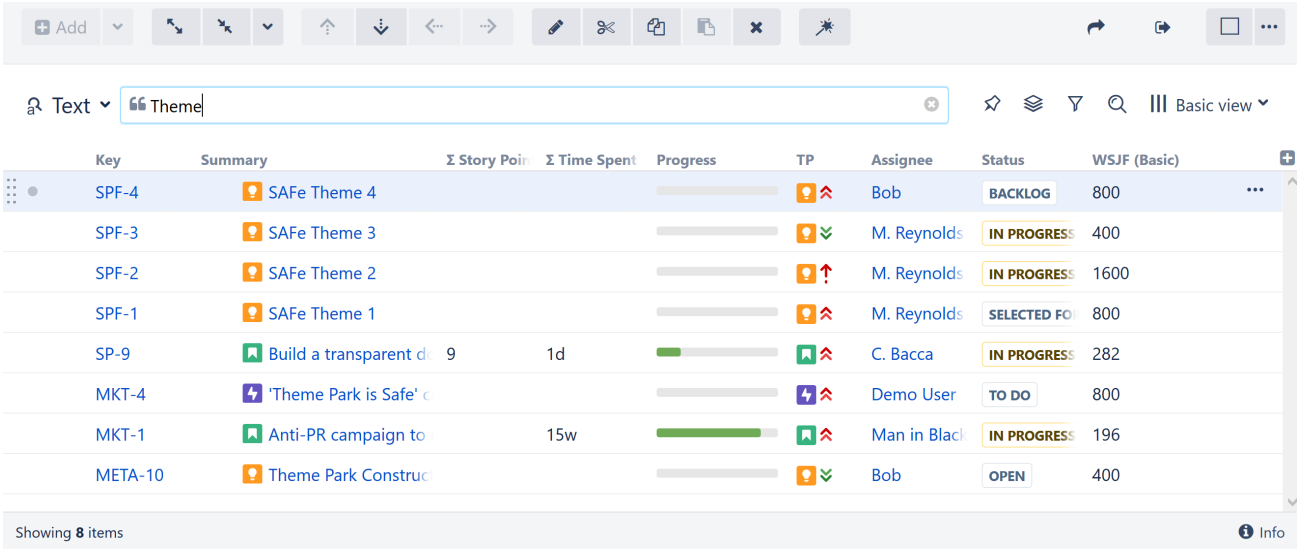
On the Structure Board you can use the structure widget not only for showing structures, but also for searching existing issues (using JQL or text search) and displaying your clipboard contents.

To search existing issues:

1. Make sure you have a structure open
2. Click the structure's name in the main panel
3. Look for the Tools section in the drop down
4. Select either Text Search or JQL Query



Once the search is open, just start typing and the results will be updated.



Just like when you're working with a structure, you can select a specific **view** for your search results and then add and arrange columns(see page 501) as necessary.

The **structure panel toolbar** also works for search results the same way it works for structures. You can apply sorting(see page 86), additional filtering(see page 207) and more complex transformations(see page 215).

⚠ Search only looks for issues from structure-enabled projects(see page 954).

3.1.3 Jira Pages with Structure

There are several pages where you can view and manage structures within Jira:

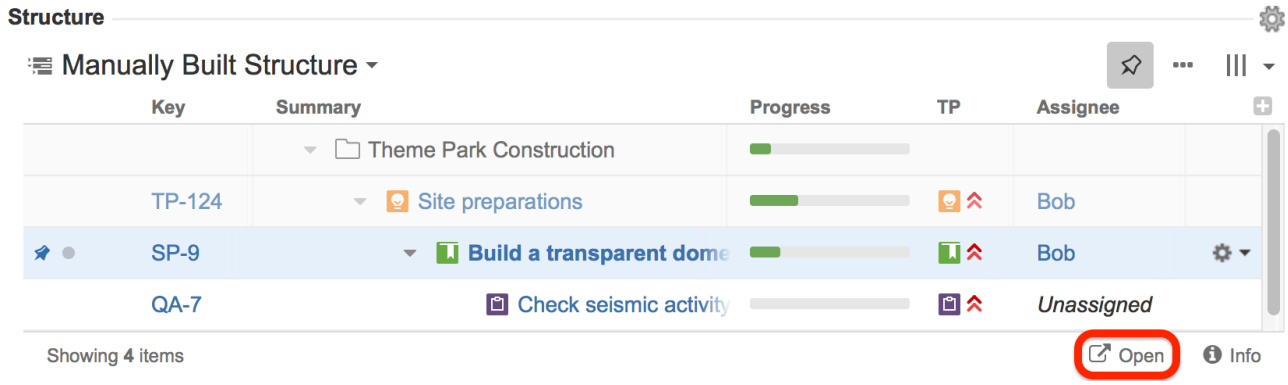
- On a dedicated Structure Board
- On the issue page
- On project pages and agile boards
- On a dashboard

Most functionality is available in each of these locations; however, since each serves its own purpose, there are some differences in behavior and appearance:

- Structure Board(see page 88)
- Structure on the Issue Page(see page 90)
- Structure on the Project Page(see page 95)
- Structure on Agile Boards(see page 96)
- Structure Gadgets(see page 565)

3.1.3.1 Open on Structure Board

Working from the Structure Board provides the most unrestricted Structure experience. To get to the Structure Board from any other page with a Structure widget, click the **Open** link at the bottom of the widget. This will open the currently viewed structure on the Structure Board.



The structure will open on Structure Board with the same filters(see page 207) and transformations(see page 215) that were applied in the original gadget. For examples, if your Dashboard Gadget is configured to only show items from a specific project, sorted by Assignee, that's exactly what you'll see on the Structure Board.

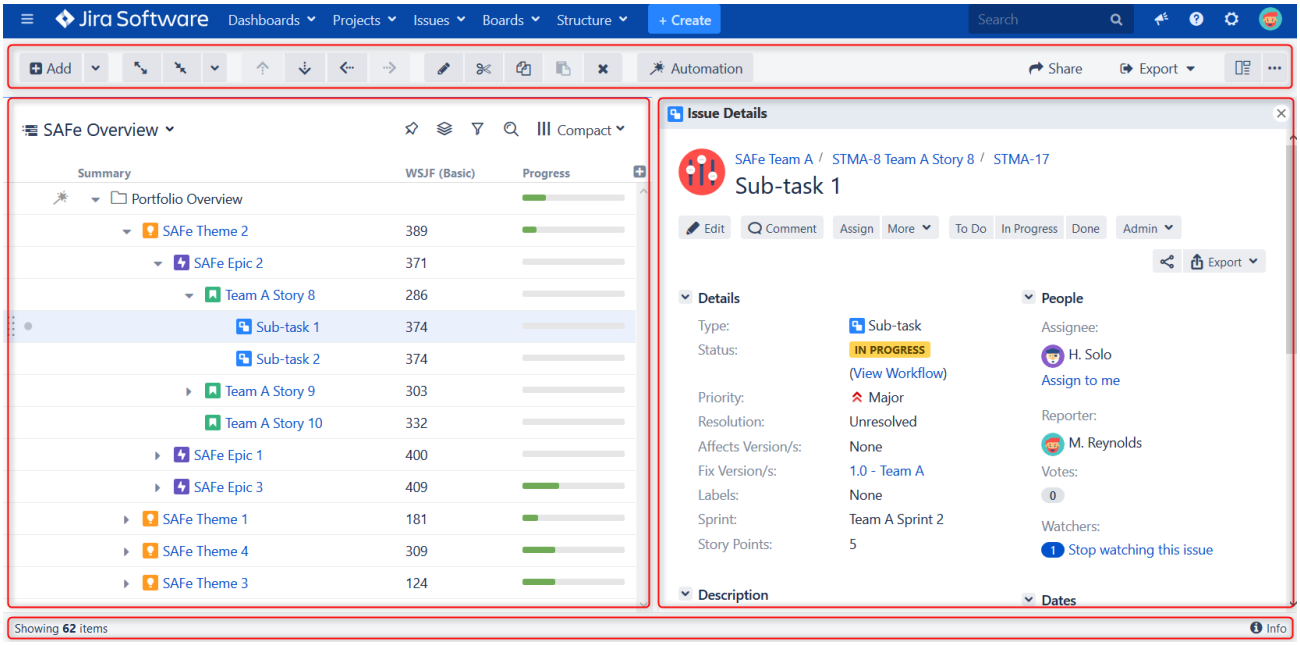
To review or remove these transformations, click the Transformations button in the panel toolbar.

3.1.3.2 Structure Board

Structure Board is a full-screen view which gives you access to all the features available in Structure.

The main elements of the Structure Board are:

- **Structure Toolbar**. At the top of the Structure Board, the Toolbar(see page 100) gives you access to the main functions for building and working with structures.
- **Working Panels.** The left panel always displays the structure widget (see page 84)or search results(see page 86), while the left panel can display another structure widget, issue details(see page 126), history(see page 556) and other features based on the add-ons you have installed.
- **Status Bar.** At the bottom of the Structure Board, this shows the number of items currently displayed, links for the **Undo** operations and notifications.



To open the Structure Board, click **Structure** in the top navigation menu in Jira and select the specific structure you want to see.

If the structure you need is not listed in the menu, there are several options:

- At the bottom of the Structure menu, select Manage Structure(see page 532). From the Manage Structure page, you can browse and search for available structures.
- Open another structure and switch between structures (see page 85)on the Structure Board.
- If you know the structure ID, you can open it directly with a URL:
  `http://your.jira.address/secure/StructureBoard.jspa?s=structure-id`

Keyboard Shortcut

Press **g** and then quickly **s** on any Jira page to open the Structure Board with the structure you opened last. (*Go Structure*)

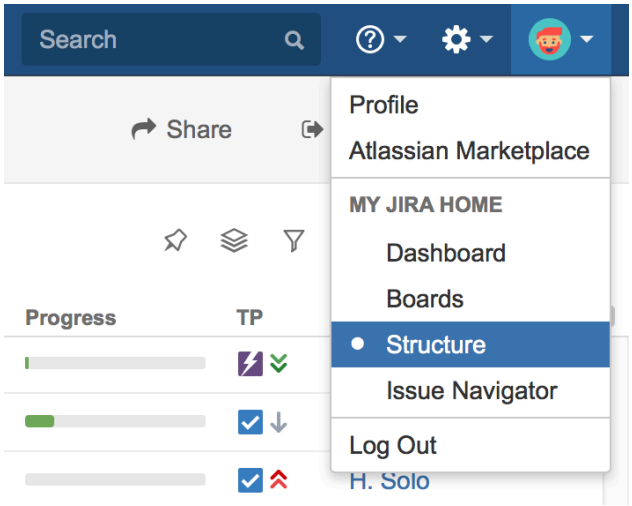
Save Time!

You can make the Structure Board your Jira Home page(see page 90).

Making Structure Board Your Jira Home

If you want to go straight to the Structure Board(see page 88) when you log in to Jira, you can make it your Jira Home page. To do so:

1. Click your avatar in the top right corner of the Jira page.
2. Select **Structure** in the **My Jira Home** section.

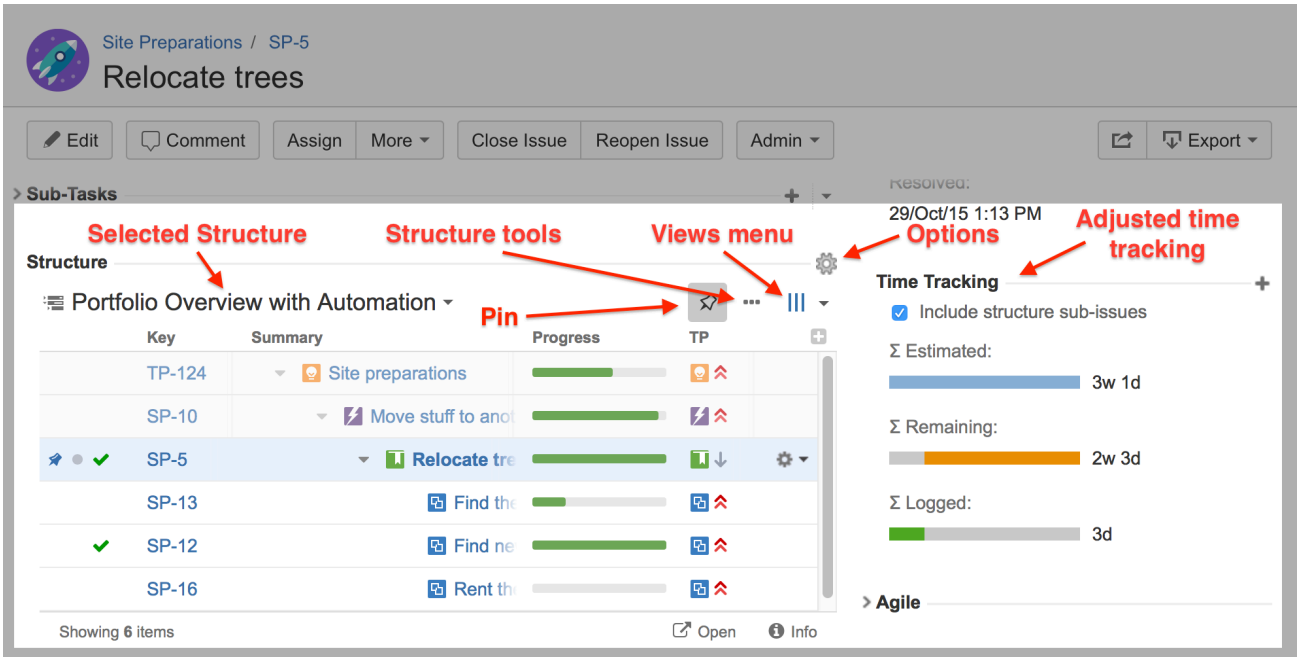


When used as a Jira Home page, the Structure Board will show your most recently opened structure.

You can also go to your Jira Home at any time by clicking the Jira logo in the top left corner of any Jira page.

3.1.3.3 Structure on the Issue Page

If an issue belongs to a project for which the Structure add-on is enabled(see page 954), the Structure widget is displayed on the issue details page. The widget is presented as a separate section, located right above the **Activity** section.



When you open an issue page, the structure that appears there is based on the Structure Options for the Issue Page(see page 93).

Pinned Issue

The issue itself is automatically located and Pinned(see page 209) in the structure. This means only the parent issues and sub-issues of the viewed issue are displayed.

You can unpin the issue to see the whole hierarchy by clicking the **Pin** button on the toolbar or by using the keyboard shortcut **ctrl + .** (period).

Structure widget can be hidden from the Issue Details page. Please refer to the Structure Administration(see page 973) article for details.

Starting with Jira 6, search results on the Issue Navigator page can display the details of a selected issue in a side panel. This details panel also contains a Structure section. Since the details panel is often much narrower than the issue page, it may be helpful to configure a view(see page 505) to fit only the necessary information in the smaller space left for the Structure widget.

Unique Features

There are several specific features on the issue page that are not present on the Structure Board:

- Collapsing/Showing Structure Section(see page 92)
- Structure Selection(see page 92)
- Adding Issue to a Structure(see page 92)
- Structure Tools(see page 93)
- Views and Options Drop-Downs(see page 93)
- Adjusted Time Tracking Section(see page 93)

- Activity Tab(see page 93)

Collapsing/Showing Structure Section

The Structure section can be hidden, as can any other section on the issue page. Once you hide the Structure section, it will remain hidden even if you open another issue page.

Also, the Structure section is automatically hidden if the issue you open does not belong to the selected structure. (This behavior can be adjusted in the Structure Options(see page 93).) When the Structure section is hidden, the issue hierarchy is not loaded from the server – it will be loaded only when you first open the Structure section.

> The *hidden* flag is stored in a browser cookie or local storage, along with flags for other sections.

Structure Selection

As you open the issue details for the first time, you will see one of the structures that contain this issue (this behavior can be adjusted through Options(see page 93)).

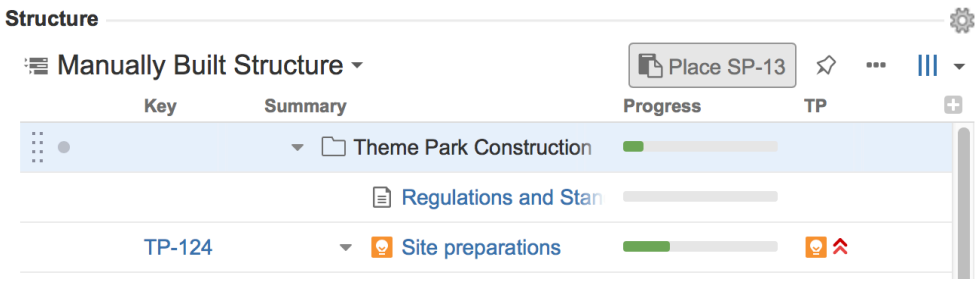
To switch to a different structure, simply click the name of the currently displayed structure and select the one you want to see. You will see those structures that contain this issue in the top section of the displayed menu.

> Structures where this issue is added through Generators(see page 140) will not appear in the list of structures containing the issue, as this would significantly affect performance.

As you switch to another structure, this new structure is memorized and shown the next time you open an issue.

Adding Issue to a Structure

If the issue you are viewing does not belong to the currently-selected structure, you can add it to this structure. To do so, unpin the issue by clicking the **Pin** button. Then select where you want the issue added (it will be added beneath whatever issue is highlighted in the structure) and click the **Place** button.



Now you can click the **Pin** button again to see only your issue and its parents and children.

> If the issue is already in the structure, you can add another instance of it to the structure using the same approach. Unpin the issue, select a location for this new instance of the issue and click the **Place** button.

Structure Tools

Next to the **Pin** button, you should see the **...** button. This allows you to access some of the basic Structure functions, including Add New issue, Expand/Collapse, Edit, Copy, Cut, Paste and Remove.

Using these tools provides much of the functionality available on the Structure Board, just in a compact form.

Views and Options Drop-Downs

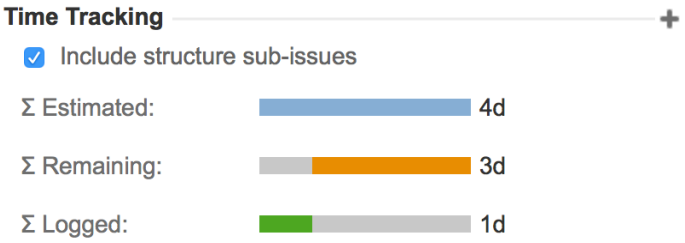
Located at the right corner of the Structure section header are Views and Options icons.

- Click Views icon to open the Views Menu(see page 507) and select another view for the displayed structure.
- Click Options icon to open Structure Options for the Issue Page(see page 93).

> An asterisk appears next to the view name if it has been locally adjusted(see page 511).

Adjusted Time Tracking Section

Structure automatically sums up time tracking information from the sub-issues and displays aggregate values in the time tracking section. Whenever any change is detected in the child issues, the time tracking information is refreshed.



You can turn off time tracking aggregation by clearing the **Include structure sub-issues** check box. The standard Jira time tracking will be shown (without Structure). The browser will remember your preference and display the original Time Tracking panel when you open other issues, until you select the **Include structure sub-issues** check box again.

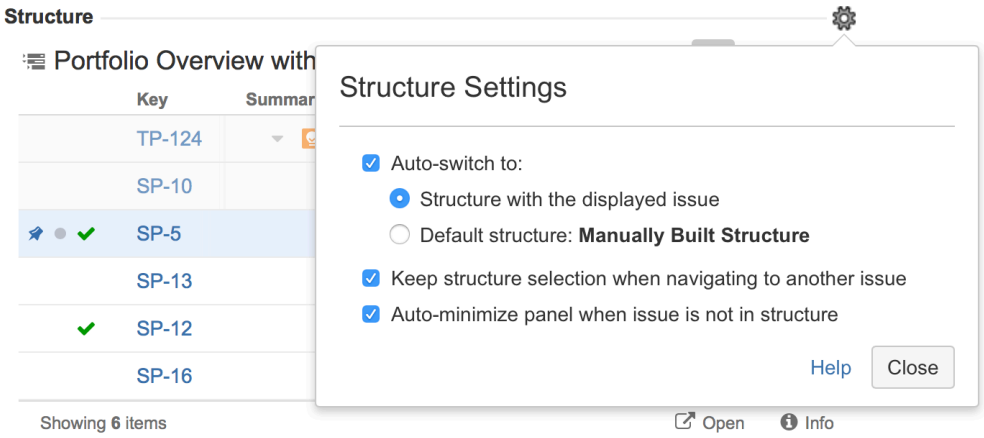
If the time tracking section is not present, it means that neither the current issue nor its sub-issues have any time tracking info.

Activity Tab

As you work with structures, all changes are added to the Jira Activity Stream. As such, all changes to the structure that affect the current issue will be displayed in the **Activity** tab of the issue page. This may be useful if you want to find out why this issue is in a particular position within a structure, including who added or moved it, and when. See Structure Activity Stream(see page 562) for more information.

Structure Options for the Issue Page

You can adjust how the Structure widget appears on issue pages. To change your Structure settings, click on the gear button in the section header. The changes are saved to the server and applied immediately.



Which Structure is Displayed?

When you have multiple structures, an issue might be present in more than one of them. When the issue page is opened, Structure needs to decide which structure to display initially.

This is controlled by a number of parameters:

Auto-switch	When auto-switch is turned on, the structure is selected based on which project and structures the issue belongs to. When auto-switch is turned off, the Structure section shows the structure that the user opened last on the Structure Board (the *current* structure).
Auto-switch: structure with displayed issue	When this auto-switch mode is selected, Structure looks for a structure that contains the issue displayed on the page. *Note: this does not include structures where the issue was added by Automation.*
Auto-switch: default structure	When this auto-switch mode is selected, the Default Structure(see page 537) for the issue's project will always be selected (even if the issue is not in that structure).
Keep structure when navigating	When you click on another issue within the Structure widget, the browser takes you to that issue's page. If this option is turned on, the new page displays the same structure as the page you navigated from (auto-switch is not applied). We recommend leaving this option on. This will prevent you from unintentionally switching structures while reviewing a structure's issues.

> ⚠ The **Keep structure when navigating** option currently does not work when you hit the **Back** button in your browser – if you return to an issue page in this manner, the structure will again be selected based on the Auto-switch settings.

Auto-Minimize?

If **Auto-minimize panel when issue is not in structure** is selected, the Structure section will be minimized if the current issue is not in the initially selected structure.

To expand the Structure section, click the section header (where it says **Structure**). You will need to click **Remove Pin** to view and edit the current structure.

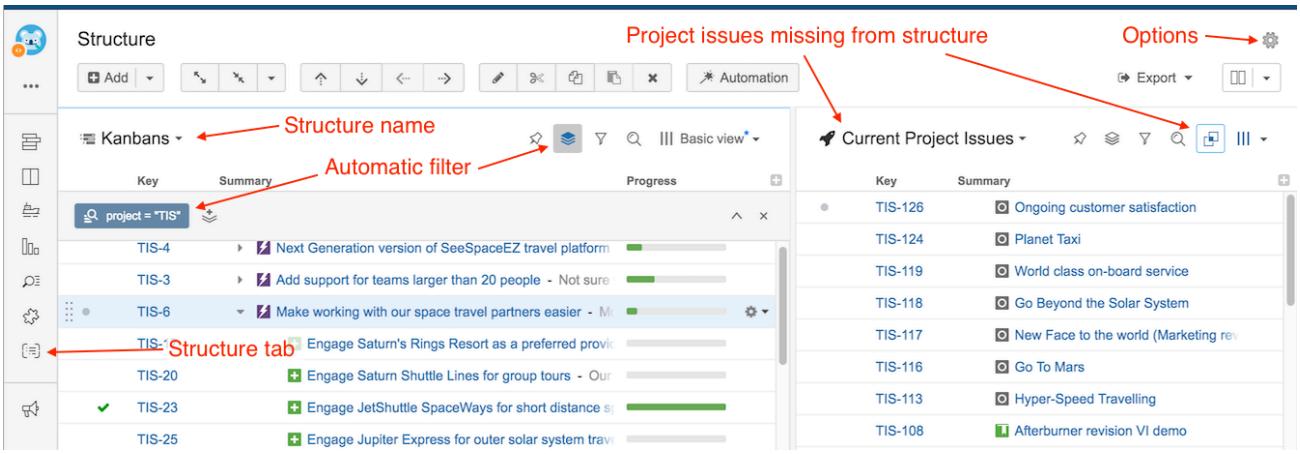
Options Scope and Default Options

When you adjust the Structure options, the changed settings apply when you view any issue on this Jira instance. (The settings are saved in your account settings.)

> ✓ The default values of these options can be configured by the Jira Administrator on the Structure Defaults(see page 957) page.

3.1.3.4 Structure on the Project Page

If a project is enabled for Structure(see page 954), you will see a new Structure icon on the side navigation bar. Clicking this will open the Structure widget on the Project Page.



This is a fully-functional Structure widget and has the same functionality you can find on the Structure Board(see page 88), with the following exceptions:

Scope

The current project defines the scope of the displayed data:

- An automatic project filter is added to the primary panel, hiding all issues from other projects. This is a non-removable transformation.
- Project issues will be displayed in the secondary panel.

If you'd like to see the full structure without the project filter, click the *Open* link in the Structure widget's footer.

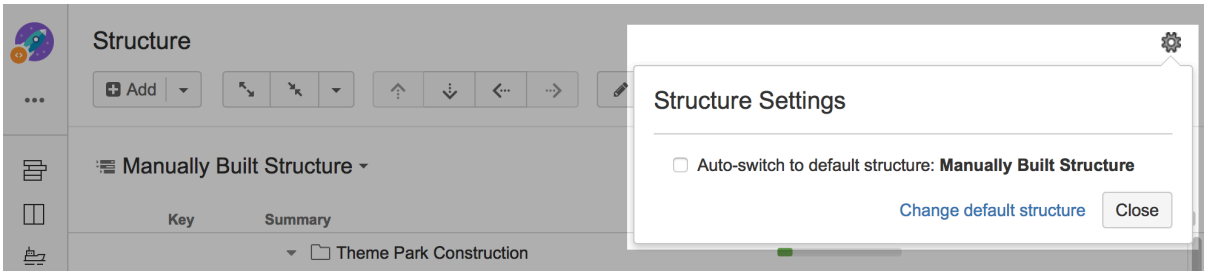
Layout

When you open the Structure tab, the Single Grid layout is selected automatically. The panel displays the most recently viewed structure or the default structure for the current project (as defined in the Options dialog). You can quickly switch to another structure by clicking the structure name and selecting the desired structure.

Project Page Options

You can make the widget open with the structure that is defined as a default structure for this specific project.

To do this, click the options gear button in the top right corner and select the **Auto-switch** option. The changes are saved to the server and applied immediately.



If you are the Project Administrator, the options dialog will also show the link to a page where you can change the default structure for your project.

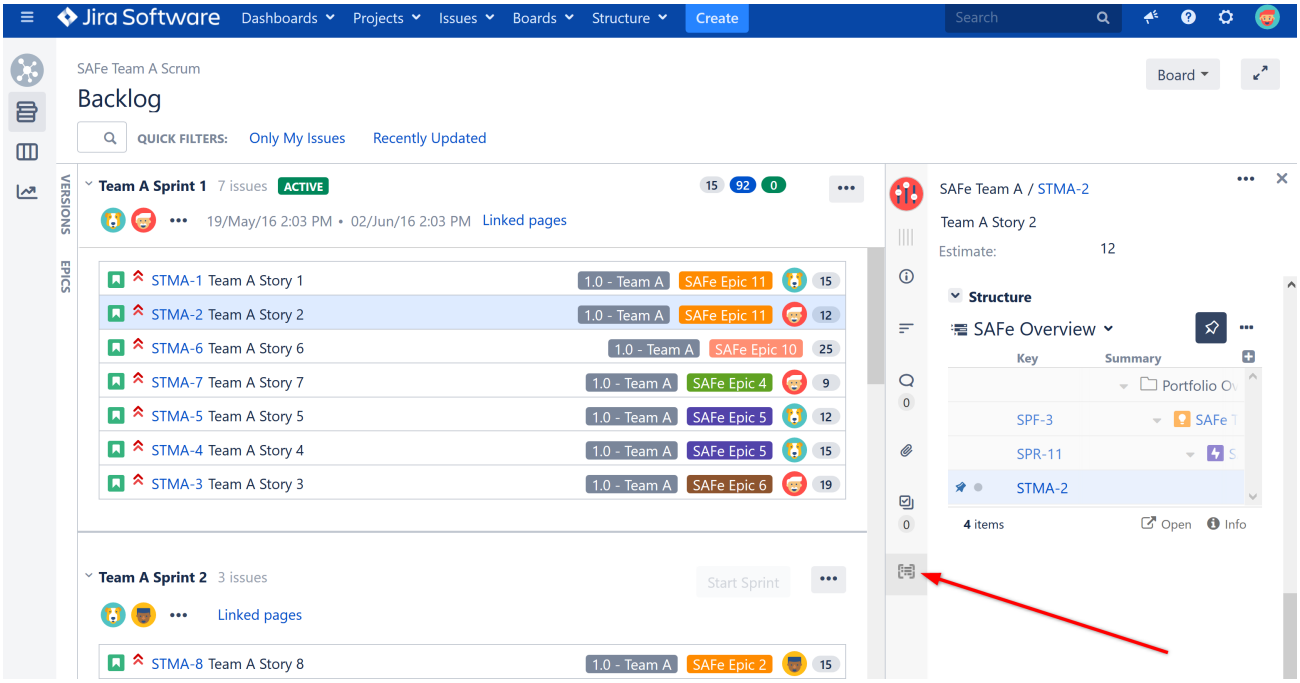
The default value for this option can be configured by the JIRA Administrator on the Structure Defaults(see page 957) page.

Perspectives are Unavailable

It is not possible to share a perspective(see page 97) from the project page.

3.1.3.5 Structure on Agile Boards

If you are using Jira Software (formerly Jira Agile or GreenHopper), you will see an additional Structure tab in the issue details panel on Scrum and Kanban boards.



The Structure tab displays the standard Structure widget in Pinned Item Mode(see page 209) (highlighting the position of the selected issue and its sub-issues). You can unpin the structure by clicking the Pin button in the toolbar or hitting **Ctrl+.** (period).

Adding Columns

Due to the constrained horizontal space, Structure initially displays only the Key and Summary columns in the Agile tab. You can add more columns(see page 501) by clicking the Add Columns button (the plus sign).

Switching Issues

When you click another issue on an Agile board, the Structure widget automatically selects and pins that issue in the current structure. You can switch to a different structure clicking the structure name.

Editing Issue Fields

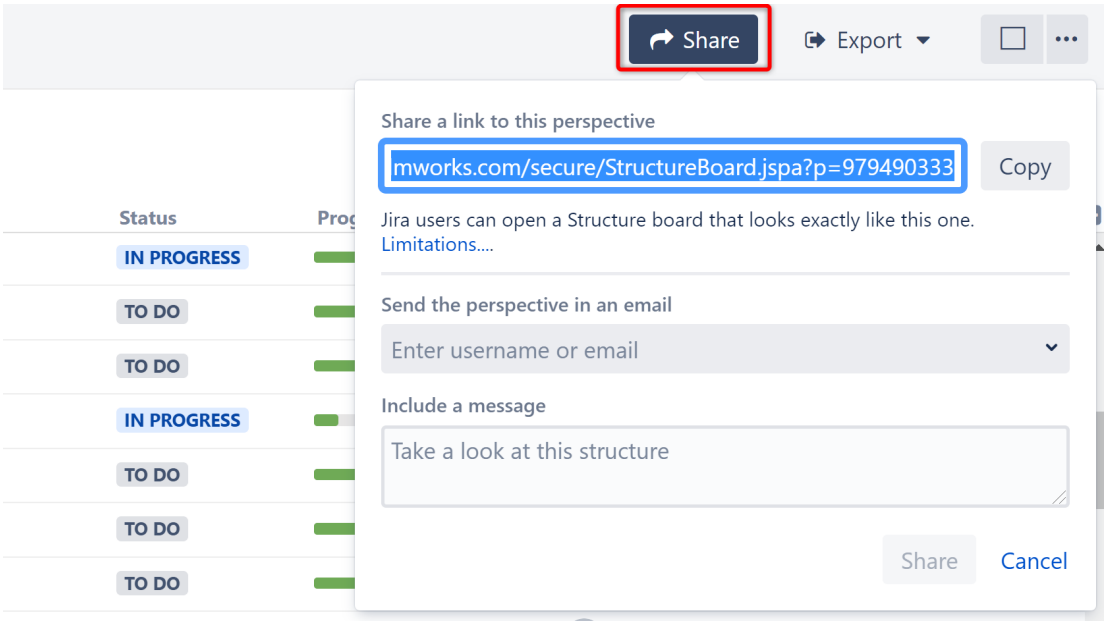
You can edit any field from within Structure widget (just as you can on the Structure Board), provided there's enough space. After you edit an issue, Structure signals Jira to reload the page, and you will see the updated values on the board.

> Only the users who have access to Structure(see page 955) will see the Structure tab on Agile boards.

3.1.4 Sharing a Perspective

Perspectives allow you to save or send a link to the current structure and its current view. Using a perspective link, users can see the structure and column layout exactly as you see it (with some limitations(see page 98)).

To save or share a perspective, open Structure Board(see page 88) and click the **Share** button in the Structure toolbar.



You can save a link to the perspective or email it to other users:

- To save a link to the perspective, click the **Copy** button.
- To email the perspective link to other users, enter each user's email address, full name or username, type a personalized message (optional), and click **Share**. This will email the perspective link AND give the users View-level access (see page 543)to the structure (if you have Control-level access).

> If the recipient already has a higher-level access, they will continue to have that access level.

3.1.4.1 Viewing a Shared Perspective

To view a perspective, simply follow the link you saved or received in an email. This will open the structure in Perspective mode - that is, the Structure Board will look mostly the way it looked when the perspective was created.

> A special **Perspective** View(see page 97) will be automatically selected. It represents the column configuration that was in use when the perspective was created. It is temporary and read-only. You can modify it, make it permanent by saving it under a new name, or switch back to some other view.

> If you receive an error message when trying to view a shared perspective, it is likely because the perspective link has expired. Contact the user who shared the link with you, and ask them to share it again.

3.1.4.2 Limitations

When creating a perspective, please consider the following:

1. If you share the link with someone who has no access to Structure, or to the individual structure for which the perspective was created, they won't be able to use your link. *Note: if you use the email sharing option, users are automatically given view-level access to the structure.*

2. If the structure contains issues accessible to you but not to the recipient, they will not see them in the structure, even in Perspective mode.
3. Issue hierarchy is not stored in the perspective. If issues have been moved around the structure or rearranged in some other way since the perspective was created, users will see those changes when they open the perspective.

   > If you need to preserve the issue hierarchy, open the structure history(see page 556) and select the latest change before creating the perspective. This way, users will see the structure in history mode when they open the perspective.

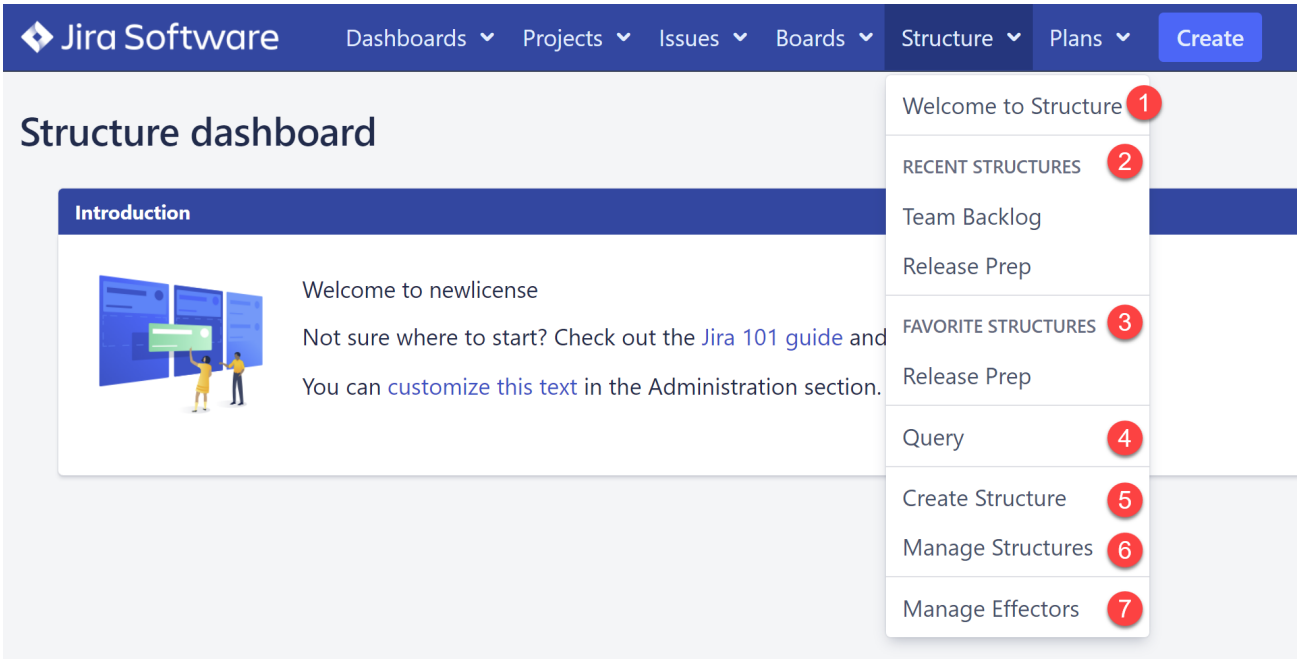
4. If a perspective is not accessed for some time, it may be automatically removed from the system. This behavior can be configured or disabled by Jira administrators via Structure Maintenance(see page 966).
5. Once created, a perspective becomes accessible to any person who has access to the structure for which that perspective was created.

3.2 Basic Operations

Structure makes it easy to view, arrange and modify issues using using the Structure Toolbar(see page 100), keyboard shortcuts(see page 116) and simple drag-and-drop functionality.

3.2.1 Structure Menu

Once you install Structure, you will see a new Structure menu added to the Jira navigation bar.

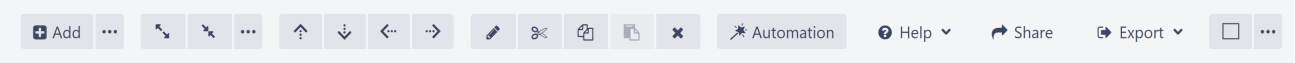


The Structure menu contains several options, some of which may not be available if you've just installed Structure:

1. **Welcome to Structure.** Provides an overview of Structure benefits, teaches you how to build structures for many popular use cases, and offers helpful links to additional resources.
2. **Recent Structures.** Shows structures that you've visited recently, or those which have been recently updated. Click a structure's name to open it.
3. **Favorite Structures.** Lists structures you have marked as your favorite. This section will be hidden if you don't have any favorite structures.
4. **Query.** Allows you to search Jira issues based on a query or text search.
5. **Create Structure.** Creates a new structure.
6. **Manage Structures.** Takes you to the Manage Structures(see page 532) screen, where you can view all the structures you have access to, search for structures, set your favorite structures, and edit the configurations for existing structures (if you have the appropriate permission).
7. **Manage Effectors**. Opens the Manage Effectors(see page 193) page, where you can review all the Effectors(see page 179) you've run and any changes they've made.

3.2.2 Structure Toolbar

The Structure toolbar provides access to the main functions of the Structure widget.

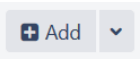
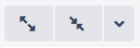
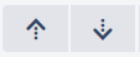


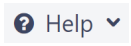
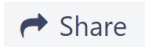
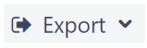
As you move the mouse pointer over the buttons in the toolbar, the active buttons are highlighted. Some buttons in the toolbar may be grayed out, indicating that these actions are not currently possible. For example, you cannot use the Paste action unless you have items in your clipboard, so the button will remain light gray and not clickable until you add items to your clip board.

> Not sure what a button does? Hover the mouse pointer over the toolbar button for a few seconds and a tooltip will appear.

3.2.2.1 Available Actions

The following actions are accessible from the Structure toolbar.

Button	Action	More Information	Keyboard Shortcut
	Create a new item and add it under the item currently selected in the structure. By default, you can add either new issues or new folders (Confluence pages are available if you have Structure.Pages installed). To search for existing issues, click the drop-down menu next to the Add button.	Creating new items	**Enter**
	Expand or collapse the whole hierarchy. Use the drop-down menu to expand the hierarchy to a specific level.	Navigating Structure(see page 82)	**++** / **--**
	Move an item up or down within the structure, without changing the item's parent. If it is not possible to move an item up or down without changing its parent, the respective option will be grayed out.	Moving Issues Within Structure(see page 108)	**Ctrl+Up** / **Ctrl+Down** or **Command+Up**/ **Down**
	Unindent / Indent the item one level. If either or both of these moves are not possible, the option(s) will be grayed out.	Moving Issues Within Structure(see page 108)	**Ctrl+Left** / **Ctrl+Right** or **Command+Left**/ **Right**
	Edit the currently selected issue / stop editing and save changes.	Editing Issues from Within Structure(see page 132)	**Tab**
	Cut the selected item(s) to the clipboard(see page 520).	Issue Clipboard(see page 520)	**Ctrl+x** or **Command+x**
	Copy the selected item(s) to the clipboard(see page 520).	Issue Clipboard(see page 520)	**Ctrl+c** or **Command+c**

Button	Action	More Information	Keyboard Shortcut
	Paste item(s) from the clipboard(see page 520) into the structure.	Issue Clipboard (see page 520)	**Ctrl+v** or **Command+v**
	Remove the currently selected item(s) from the structure.	Removing Issues(see page 115)	**Delete**
	Switch Automation editing mode on/off.	Generators(see page 140)	~
	Access help rescources, including the Welcome to Structure guide, documentation, and more.		
	Create a perspective(see page 97) link to share the current view.	Perspective(see page 97)	
	Open a printable page(see page 561) with the structure or export the structure to Excel(see page 558).	Printing(see page 561) Exporting to Excel(see page 558)	
	Select the layout options, including opening the secondary panel(see page 518).	Double Grid Mode(see page 518)	

3.2.3 Adding Items to a Structure

The following types of items can be added to a structure:

- Existing Issues(see page 102)
- New Issues(see page 128)
- Folders(see page 104)
- Memo(see page 105)
- Generators (see page 140)
  (Automation)(see page 140)
- Effectors(see page 179)

Click any of the above to learn more about the item type and how to include them within a structure.

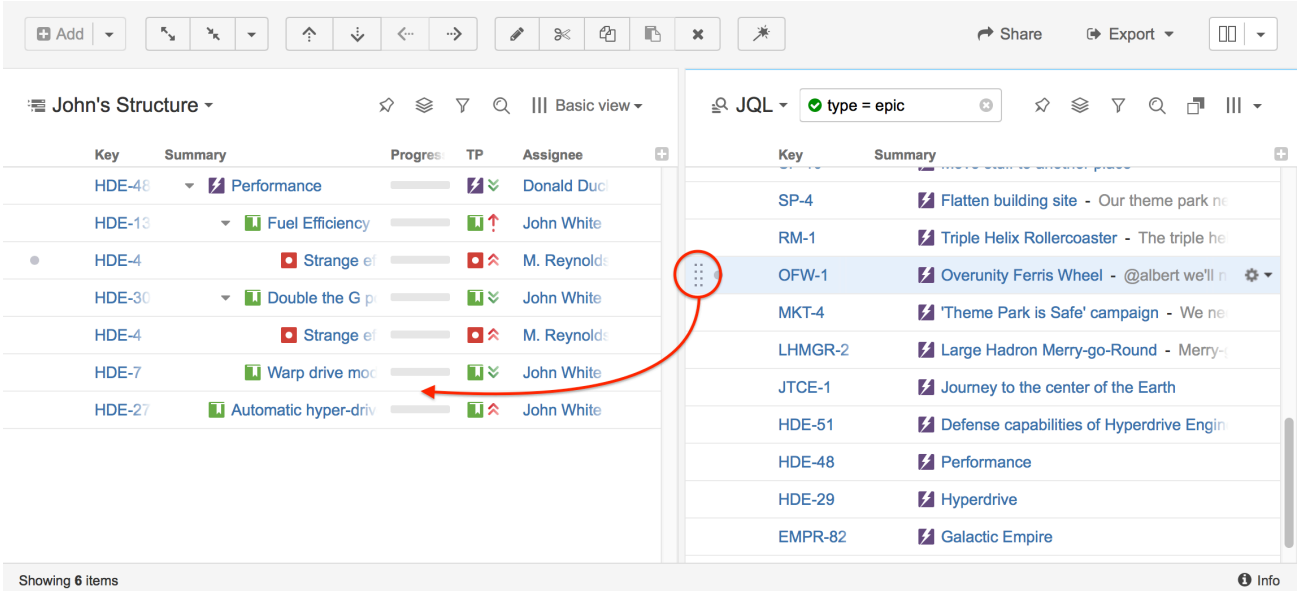
3.2.3.1 Adding Existing Issues

Issues can be added to a structure from the Structure Board(see page 88) or their own Issue Page(see page 92).

Adding issues from Structure Board

On the Structure Board, open the secondary panel(see page 518) and search for existing issues using the text or JQL search. You can import issues one at a time, or select multiple items(see page 113).

After selecting the issue(s) you need, add them to your structure using drag-and-drop(see page 109) or copy/paste(see page 110).

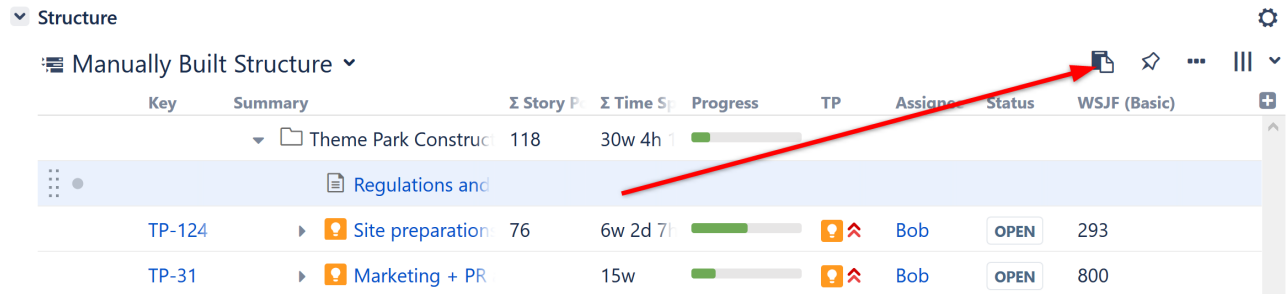


You can also add issues from another structure. Simply open the structure in the secondary panel(see page 519), and use drag-and-drop or copy/paste to import issues.

Adding issues from the Issue Page

You can also add issues to a structure directly from their issue page.

From the issue page, select the structure you want to add the issue to, highlight the place you would like to insert the image (it is inserted immediately after the highlighted line) and click the Paste button.



Issues can be added and removed automatically using Generators(see page 140).

3.2.3.2 Folders

Folders can be added to a structure to group issues within your hierarchy.

Some common uses for folders include:

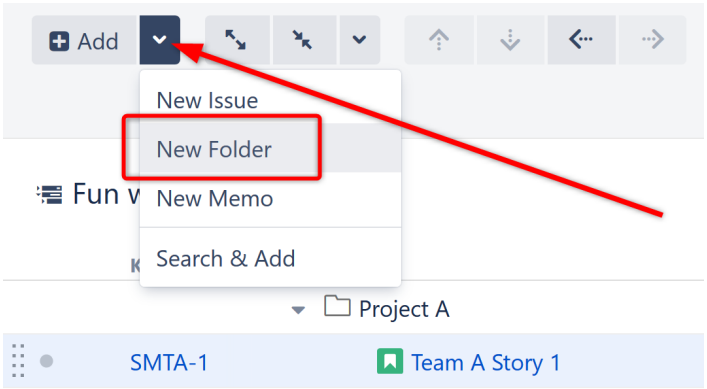
- Organizing issues into specific categories
- Separating different projects or different parts of a project
- Creating different Generators(see page 140) rules for different parts of a structure (if a generator is placed beneath a folder, it will only affect items in that folder - see Generator Scope(see page 198))
- Placing a structure within a structure (it's not necessary to use folders, but we recommend it)

> Group generators(see page 169) make their own folders to group items by a common attribute.

Adding Folders to a Structure

To add a folder to a structure:

1. Select the folder's location (the folder will be placed at the same level in the hierarchy, beneath the currently-selected item)
2. Open the **Add** drop-down menu
3. Select **New Folder**

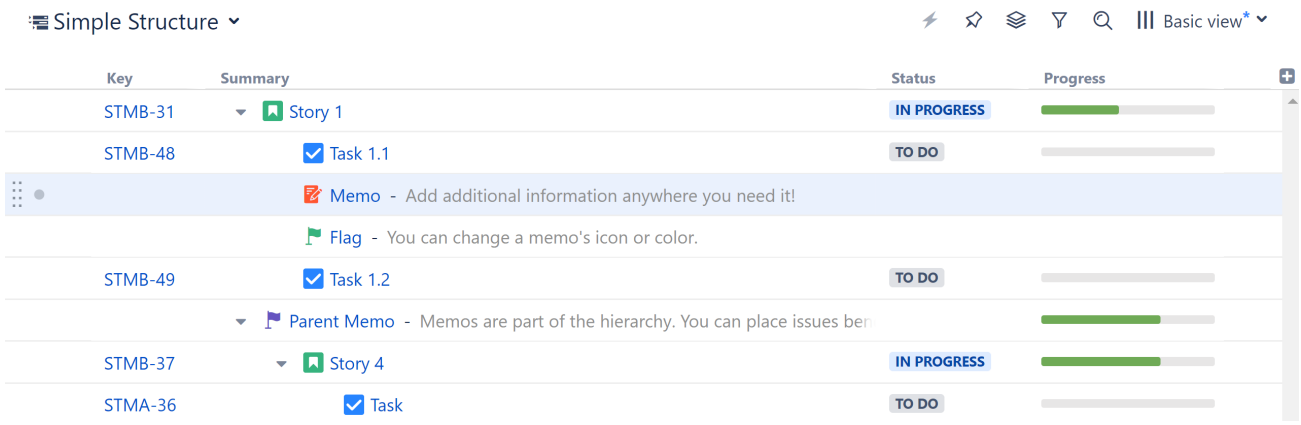


Keyboard shortcut: Press **Enter** to open the Add New Item dialogue, and press **Alt+Up/Down** to select between Issue, Folder and Memo.

Users must have Edit permissions(see page 543) or higher to add folders to a structure.

3.2.3.3 Memo

Memos work similar to folders within a structure, except that memos can include a choice of icons, color and rich text.



Memos can serve a variety of purposes within a structure:

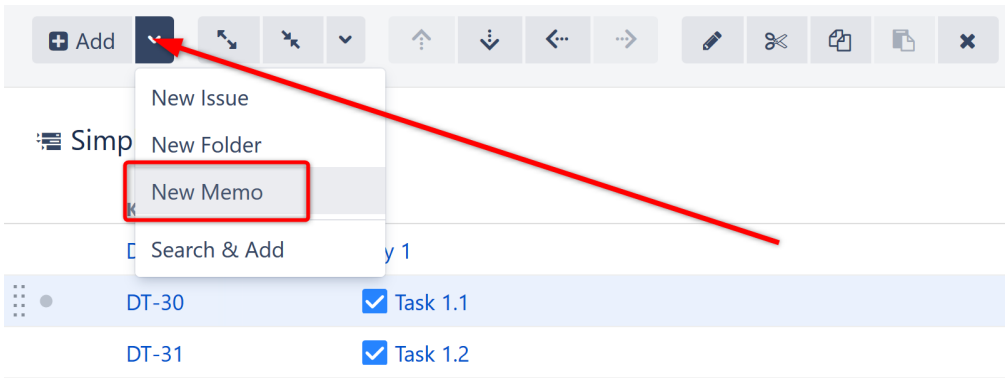
- Add notes or reminders that pertain to the structure or project as a whole, rather than just a single issue (for that, try a Notes column(see page 470))
- Add high-level requirements directly to your structure
- Add a placeholder for other items
- Use them like folders, grouping issues within your hierarchy (with the added benefit of color and text)
- Just about anything else you can think of!

Adding a Memo to a Structure

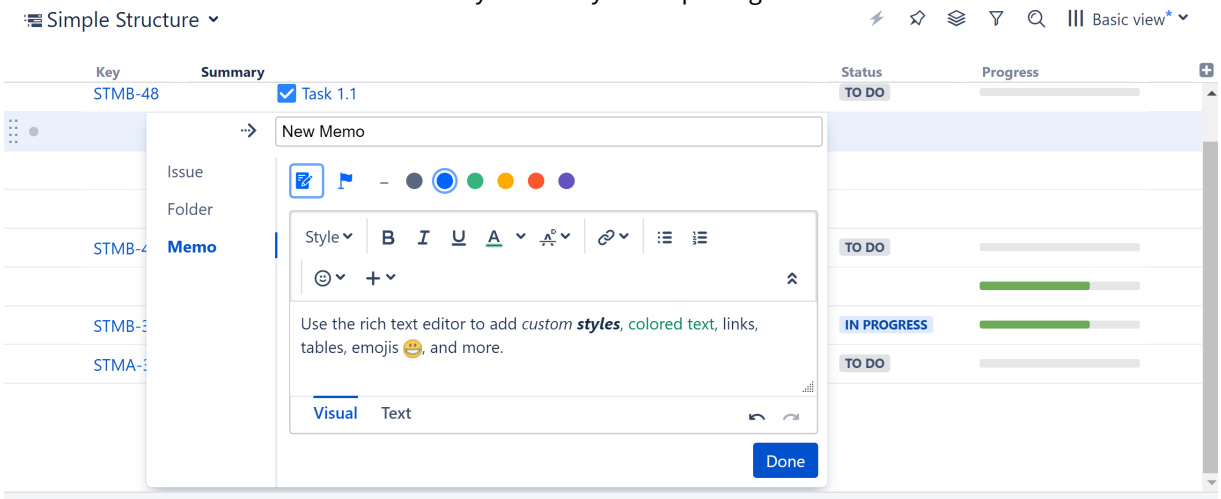
To add a memo to a structure:

1. Select the memo's location (the memo will be placed at the same level in the hierarchy, beneath the currently-selected item).
2. Open the **Add** drop-down menu.

3. Select **New Memo.**



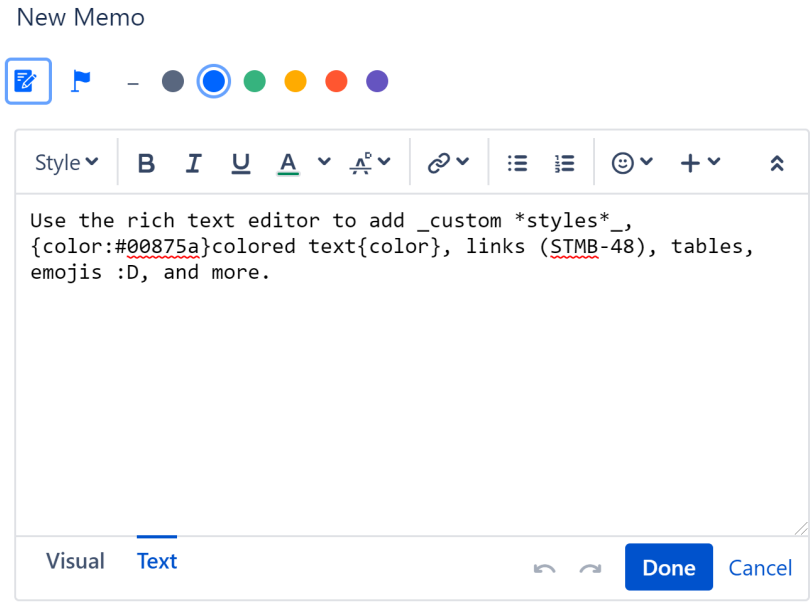
4. Choose the icon and color. Add text and any necessary markup using the built-in rich text editor.



The editor opens in Visual mode, a WYSIWYG editor. You can also edit text using wiki markup by switching to the Text mode - see Jira's Text Formatting Notation Help page[21] for a complete list of available formatting

21 https://jira.atlassian.com/secure/WikiRendererHelpAction.jspa?section=all

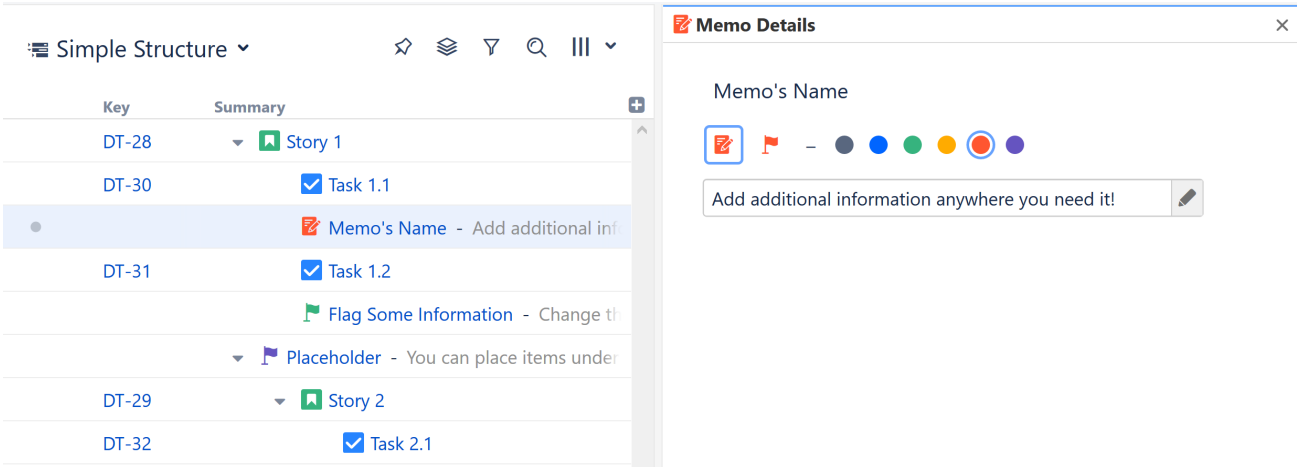
options and conventions.



Keyboard shortcut: Press **Enter** to open the Add New Item dialogue, and press **Alt+Up/Down** to select between Issue, Folder and Memo.

Editing a Memo

Once a memo is placed in the structure, you can easily change its icon, color or text by clicking on the memo's name within the Summary column. This will open up the Memo Details panel, where you can make any necessary changes.



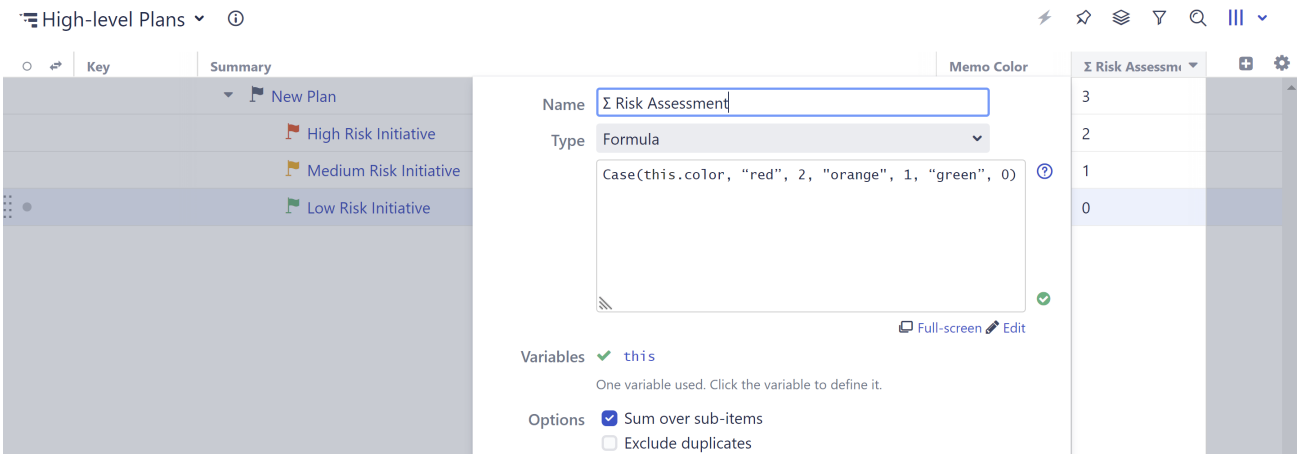
It is not possible to edit a note from the Description column.

The memo name can also be edited directly from the structure by clicking the edit icon in the toolbar or using the **Tab** or **F2** keyboard shortcut.

Memo Values in Formulas

Memo values for color, description, and title can be used in formulas with the following formats:

- this.color
- this.description
- this.title



In this example above, we assigned a "risk" value to each memo color and aggregated those values across our plan.

Some common uses for memo values in formulas include:

- Using memo colors to identify risk levels in your planning stage and aggregating risk across the structure (see above)
- Using key words in memo titles or descriptions to narrow your formula results
- Grouping issues under memos and creating different calculation rules based on the parent memo values

3.2.4 Moving Items within Structure

Using the Structure toolbar or your keyboard, you can move items up or down within a structure or change their location within the hierarchy.

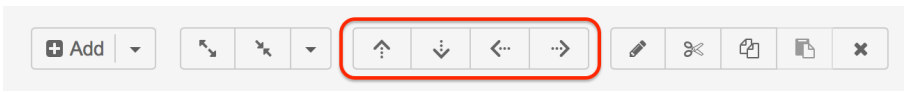
Items can be moved in any of these ways:

- Using the Structure toolbar(see page 108)
- With the mouse (Drag and Drop(see page 109))(see page 0)
- Via cut/copy and paste(see page 110)
- Using keyboard shortcuts(see page 116)

3.2.4.1 Moving Items with the Toolbar

Using the Structure toolbar or your keyboard, you can move items up or down within a structure, or change their location within the hierarchy, one position at a time.

To move an item, simply highlight it in your structure and use one of the following commands.



Operation	Keyboard Shortcut	What it does
Move Up	Ctrl + Up	Without changing the item's parent, moves the item up and places it before the previous child - if possible.
Move Down	Ctrl + Down	Without changing the item's parent, moves the item down and places it after the next child - if possible.
Level Up / Outdent	Ctrl + Left	Move the item one level up. This will place the item after its parent.
Level Down / Indent	Ctrl + Right	Move the item to be a sub-issue of its current preceding sibling.

Mac Users: Use Cmd instead of Ctrl.

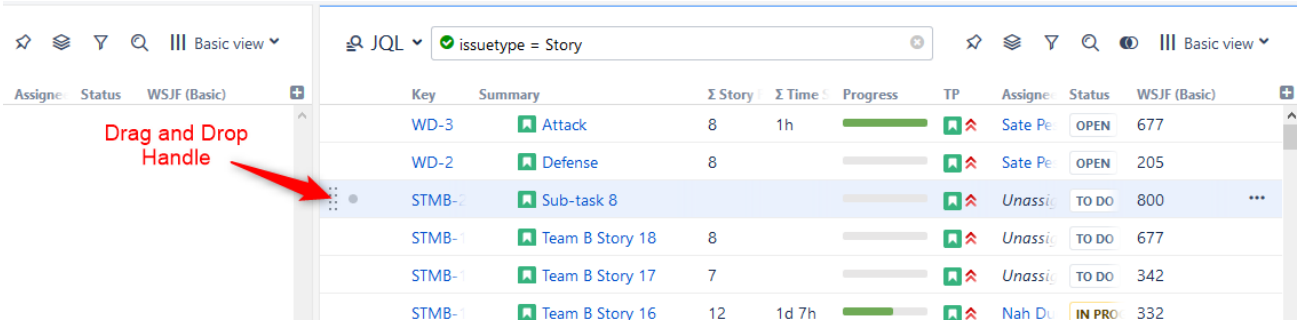
⚠ When you move an item that has sub-items, the whole sub-tree is moved.

3.2.4.2 Drag and Drop

Drag and drop allows you to quickly move or copy items within the structure or add items to a structure using the secondary panel(see page 518).

Moving Items

To grab an item, move your mouse pointer over the Drag and Drop Handle, at the far left of the item's row. Press and hold the handle as you move the issue to it's new location.



As you move the item over the grid, the structure will rearrange itself to show the possible positions for the dragged items. Once the item is in the correct place, release the mouse button and the item will be moved.

Depending on how you move an item, a couple of different things can happen:

- **Moving Up or Down** - Moves items up and down the hierarchy without changing the indentation level, if possible.
- **Moving Left or Right** - Changes indentation level of the moved items, if possible.

> Holding the **shift key** allows you to grab an item from anywhere on the row (except on a link). This is especially useful if you need to outdent an item, since the drag handle is usually close to the edge of the screen.
>
> After dragging has started, you can release the shift key.

Copying Items

To copy an item within the same structure (creating a second instance of the item), hold the **Ctrl** key as you drag it (**Alt** key for Mac).

To copy an item from one structure to another (using Double Grid Mode(see page 518)), simply drag and drop the item. If you want to move the item from one structure to another (so it's removed from the original structure) hold the **Ctrl** key as you drag it (**Alt** key for Mac).

Dragging Multiple Items

To move more than one issue at a time, first select multiple issues(see page 113) (click the gray dot at the beginning of each row) and then move them using the Drag and Drop Handle of any of the selected items.

> If you have multiple items selected, but start dragging an item that's not included in the multiple selection, only that item is dragged.

Cancelling Drag

If you need to cancel drag and drop without moving the item, press the **Escape** key.

> Drag and drop can also be undone(see page 115).

Scrolling Structure While Dragging

To move an item to a location not visible in the current panel, drag the item to the top or bottom edge of the structure widget and the structure will be scrolled up or down. The further you move the dragged item, the faster the the screen will scroll.

> Using Cut & Paste(see page 110) may be more effective than Drag and Drop if you need to move items to a distant position.

3.2.4.3 Cut, Copy and Paste

You can easily move items using Cut, Copy and Paste. This is particularly useful when moving an item across a large structure.

First, select the issue by clicking anywhere in the issue's row or using up/down on your keyboard. The selected row will be highlighted.

- **To move the item**, click the **Cut** button  in the toolbar. Then highlight the new location you want the item and click the **Paste** button  . The original item will be placed in the row below the selected item.
- **To create a copy of an item**, click the **Copy** button  . Then highlight the location you want the copy placed and click **Paste**  . The copy will be placed beneath the highlighted item.

When you copy or cut issues using the toolbar or Ctrl+C / Ctrl+X (Command+C / Command+X for Mac), the selected items are placed into the clipboard. When you paste with Ctrl+V (Command+V), items are added immediately **after** the currently focused item.

To paste items **under** the currently focused item (as its children), press Ctrl+Shift+V (Command+Shift+V).

You can also cut/copy and paste multiple issues at one time. See Selecting Multiple Items(see page 113) for more details.

Copy / Paste Scenarios

There are two main scenarios for using the Issue Clipboard(see page 520):

- Moving Items Between Structures(see page 111)
- Moving Items Within a Structure(see page 112)

If you have any text selected on the page, the **Copy/Cut** keyboard shortcuts will operate on that text, not the items in your structure. The text will be copied to the system clipboard, and the Structure clipboard will not be affected.

Moving Items Between Structures

The contents of the Structure clipboard are preserved in the **current browser window**, which allows you to copy/cut items in one structure and paste them into another.

To copy items (with their sub-items) from one structure to another:

Step 1 - Cut/Copy

First add the desired items to the clipboard:

1. Open the structure you wish to cut/copy from.
2. Select the items you want to cut or copy. Either select a single item, or use multiple select(see page 113).
3. Click the **Cut/Copy** button on the structure toolbar (or press **Ctrl+x/Ctrl+c** or **Command+x/Command+c**).
4. The selected items will be added to the clipboard and marked with a small scissors icon for cut  or a clipboard icon for copied  .

Cut items are not removed from the structure until you paste them into another structure.

When you **copy** an item that contains sub-items, these sub-items are not automatically copied with their parent. You need to select them explicitly. To copy an item and all its children:

1. Select a parent item
2. Press Shift+Arrow Right (this selects an item and all of the sub-issues)
3. Press the Copy button

When you **cut** an item, its sub-items are automatically cut and copied to the clipboard.

Step 2 - Paste

After you have cut/copied the items, you can now paste them into any other structure:

1. **In the same browser window**, switch to a desired structure (you can use Structure Board or any other Jira page with Structure(see page 88)).
2. In the structure grid, select the item after which the items from the clipboard should be placed.
3. Click the **Paste** button on the toolbar (or press **Ctrl+v** or **Command+v**) to place the items at the same indentation level. To place the items **under** the selected item (as its children), press **Ctrl+Shift+v** (**Command+Shift+v** on Mac).

If any of the items you are pasting are already in the new structure, those existing items will not be affected. New copies of the items will be created as you paste.

The Paste operation can be undone(see page 115).

If you need to copy the same set of issues to several different structures, you can open the Issue Clipboard(see page 520) in the secondary panel(see page 518) and use drag-and-drop(see page 109) to move the issues, instead of Paste. In this case, the issues will not be removed from the clipboard.

Moving Items Within a Structure

You can use Copy/Cut and Paste to move items within a structure. This is particularly useful when moving several items across a large structure, where drag-and-drop(see page 109) can be more difficult to manage.

To move an item within a structure:

Copy/Cut

First add the desired items to the clipboard:

1. Select the items you want to move. Either select a single item, or use multiple selection(see page 113).
2. To move the item, click the **Cut** button on the structure toolbar (or press **Ctrl+x** or **Command+x**). To create a second copy of the item within the structure, use the **Copy** button (or **Ctrl+c** or **Command+c**).
3. The selected items will be added into the clipboard and marked with a small scissors icon (for cuts) or a clipboard icon (for copies).

Cut items are not removed from the structure until you paste them to a new location.

When you **copy** an item that contains sub-items, these sub-items are not automatically copied with their parent. You need to select them explicitly. To copy an item and all its children:

1. Select a parent item
2. Press Shift+Arrow Right (this selects an item and all the sub-issues)
3. Press the Copy button

When you **cut** an item, its sub-items are automatically cut and copied to the clipboard.

Paste

After you have cut/copied issues, you can paste them to any place in the structure:

1. To view the contents of your **Clipboard**, you can open it in the secondary panel(see page 518). However, this is not necessary.
2. In the structure, select the item after which the items from the clipboard should be placed.
3. Click the **Paste** button on the toolbar (or press **Ctrl+v** or **Command+v**) to place the items at the same indentation level. To place the items **under** the selected item (as the children), press **Ctrl+Shift+v** (or **Command+Shift+v** on Mac).

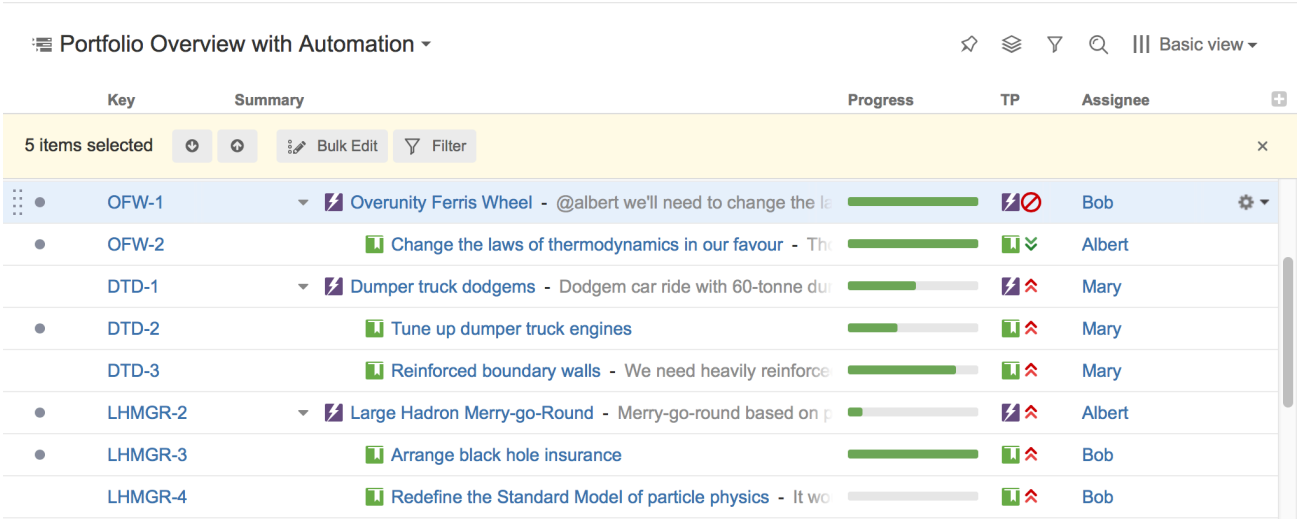
The Paste operation can be undone(see page 115).

3.2.4.4 Selecting Multiple Items

To select multiple items (for moving, copying, deleting, or editing), do one of the following:

- Click the gray dot at the beginning of the item row for each item you wish to select
- Press **Space** to add the currently-focused item, and then move to the next issue and press **Space** again
- Hold **Shift** and use the Up and Down arrows to select a range of issues
- Hold **Shift** and use the Right/Left arrows to select/deselect the focused issue with all its sub-issues
- Hit **Ctrl+A** (**Command+A** on Mac) to select all issues

Selected items are marked with a filled circle, and an additional panel appears at the top of the grid showing the number of selected items and several action buttons.



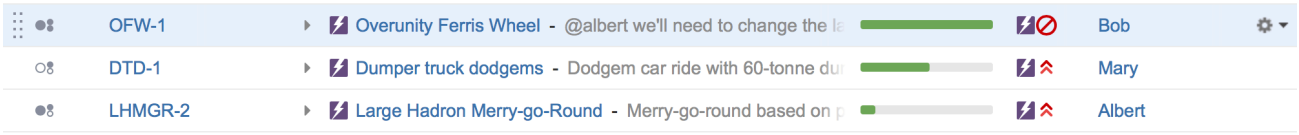
The selection panel offers the following features:

- Move focus from one selected item to another by clicking the up and down arrows
- Bulk Edit(see page 138) the selected issues using the Jira bulk change wizard
- Show only selected items and their parents by clicking the Filter button
- Remove all selections by clicking the close button in the right corner of the panel

Special Selection Markers

If you collapse a list of sub-items, the selection marker of the parent item will show if it contains any selected sub-issues.

For example, if you collapse sub-issues of *OFW-1*, *DTD-1*, and *LHMGR-2* in the example below, you will see these selection markers (the large and small circles to the left of each row):



The meaning of each marker is as follows:

	The item and **all** its sub-items are selected.
	The item and **some** of its sub-items are selected.
	The item itself **is not** selected, but **some** of its sub-items are selected.
	The item itself **is not** selected, but **all** of its sub-items are selected.

Changing Multiple Items

The following actions work with the multi-selection:

- Drag and drop(see page 109) lets you move a selection of items within a structure or between two structures
- Cut and paste(see page 110) allows you to move items within a structure and between different structures
- Remove button(see page 115) or **Delete** key lets you remove multiple items from the structure
- Toolbar buttons *Move Up*, *Move Down*, *Indent* and *Outdent* are allowed for multiple items, only if all items in the selection are at the same level in hierarchy and have the same parent item
- Bulk Change(see page 138) lets you use the Jira bulk change wizard to modify the selected issues

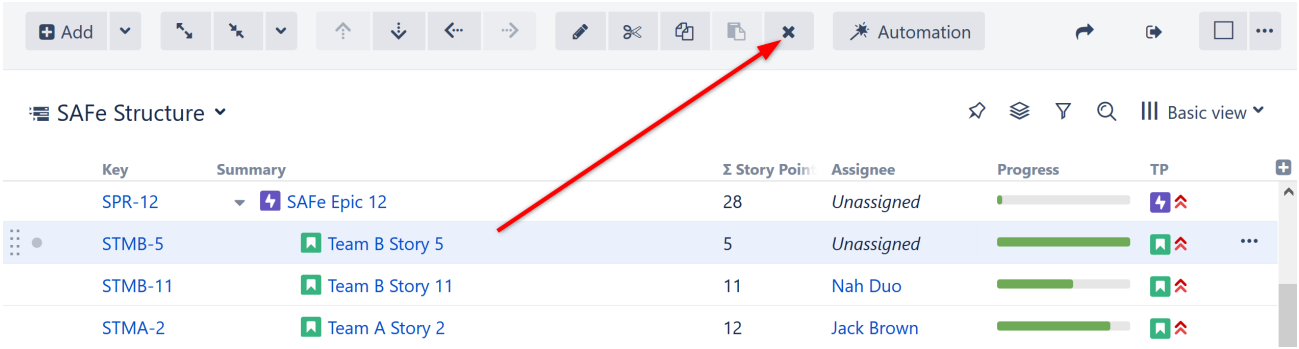
Exiting Multi-Select Mode

To exit multi-select mode (and deselect all items), press the **x** button at the far right of the selection panel or press the **Escape** key.

You can also press **Ctrl+A** (**Command+A**) twice – the first key stroke will select all items, the second one will deselect all items.

3.2.5 Removing Items from Structure

To remove an item from the current structure, select the item and press the **Delete** button (on your keyboard or in the toolbar). The selected item is removed with all its children items.



Removing an issue from a structure does not delete the issue itself. It simply removes it from the current structure.

To delete more than one item, select the items(see page 113) and click **Delete**.

Removing items can be undone(see page 115).

3.2.6 Undoing Changes

Structure lets you undo an operation if you realize that you have made a mistake or that the result is not what you expected. The following operations can be undone:

- Manually adding(see page 102) issues
- Removing(see page 115) items from a structure
- Drag-and-Drop(see page 109)
- Cut & Paste(see page 110)

When you perform an operation that can be undone, a corresponding hyperlink appears in the footer at the bottom of the Structure screen. For example, if you drag and drop some items, the link will read "Undo Drag and Drop". If you click the link, your changes are reverted, and the link itself changes to a "redo" link, allowing you to reapply the operation.

When you remove items(see page 115), a notification pop-up with an "undo" link also appears at the top of the page.

Currently only the last operation can be undone, but we are working on additional functionality for Undo.

If the operation being undone has been uploaded to the server already, then a new operation (or several operations) will be uploaded in order to revert the changes. You will see both the original operation and the undo operation in the structure history(see page 556).

3.2.7 Keyboard Shortcuts

Structure provides a number of keyboard shortcuts that you can use to speed up your work. These reference cards describe the shortcuts for PC and Mac OS X keyboards.

- Keyboard Shortcuts (PC)(see page 116)
- Keyboard Shortcuts (Mac)(see page 121)
- Quick Action Lookup(see page 125)

3.2.7.1 Keyboard Shortcuts (PC)

Navigation

Action	**Shortcut**
Select Issue	*Left-Click*
Show/Hide Issue Details	`o`
Previous Issue	↑ *or* `k`
Next Issue	↓ *or* `j`
Expand Sub-Issues	→
Collapse Sub-Issues	←
For Large Structure	`PgUp PgDn Home End`
Add Column	`tt`
Expand All	`++`
Collapse All	`--`

Structure Views

Action	Shortcut
Switch View	`vv`
Return to Previous View	`vvv`
Save View	`vs`
Save View As	`vss`
Revert Changes to View	`vr`
Text Wrapping	v w
Horizontal Scrolling	v h

Searching & Adding to Structure

Action	Shortcut
Switch Structure	`ss`
Add Issue	`Ctrl+Enter`

Standard JIRA Actions

Action	Shortcut
Operations Dialog	`.`
Edit Issue	`e`
Comment on Issue	`m`

Action	Shortcut
Assign Issue	`a`
Assign to Me	`i`
Edit Issue Labels	`l`
Actions Drop-Down	`Alt+↓`

Changing Structure

Action	Shortcut
Move Up	`Ctrl+↑`
Move Down	`Ctrl+↓`
Indent	`Ctrl+→`
Outdent	`Ctrl+←`
Drag and Drop	`Shift+Drag`
Drag and Drop (Copy)	`Ctrl+Shift+Drag`
New Issue	`Enter`
New Sub-Issue	`Insert` *or* `Shift+Enter`
Remove from Structure	`Delete`
Select between Folder/Issue/Page (in Add dialog)	Alt+↑ or Alt+↓

Changing Issues

Action	**Shortcut**
Edit Field	*Double-Click*
Edit Summary	`Tab` *or* `F2`
Finish & Save	`Enter` *or* `Ctrl+Enter`
Cancel Field Changes	`Esc`
Edit Next Field	`Tab` *or* `Ctrl+Alt+→`
Edit Previous Field	`Shift+Tab` *or* `Ctrl+Alt+←`
Edit Next Issue	`Ctrl+Alt+↓`
Edit Previous Issue	`Ctrl+Alt+↑`

Selecting Issues

Action	**Shortcut**
Toggle Selection	`Space`
Select All	`Ctrl+a`
Select All Sub-Issues	`Shift+→`
Deselect All Sub-Issues	`Shift+←`
Expand Selection Down (Up)	`Shift+↓` (`Shift+↑`)

Action	Shortcut

Bulk Selection	`Shift+PgUp`
	`Shift+PgDn`
	`Shift+Home`
	`Shift+End`

Clear Selection	`Escape`

Advanced

Action	Shortcut
Hide/Show Resolved	`rr`
Cut (Prepare to Move)	`Ctrl+x`
Paste (Move)	`Ctrl+v`
Paste Sub-Issue (Move)	`Ctrl+Shift+v`
Fix/Unfix View on Issue	`Ctrl+.`
Switch Panel	\
View Full-Size Image(see page 467)	`ii`
Show/Hide Issue Details without Switching Panel	`Shift+o`
Show Automation	~
Toggle Structure description	d d

3.2.7.2 Keyboard Shortcuts (Mac)

Navigation

Action	**Shortcut**
Select Issue	*Left-Click*
Show/Hide Issue Details	o
Previous Issue	↑ *or* k
Next Issue	↓ *or* j
Expand Sub-Issues	→
Collapse Sub-Issues	←
For Large Structure	⇞ ⇟ ↖ ↘
Add Column	tt
Expand All	++
Collapse All	–
Change Structure	xx

Structure Views

Action	**Shortcut**
Switch View	vv
Return to Previous View	vvv

Action	Shortcut
Save View	`vs`
Save View As	`vss`
Revert Changes to View	`vr`
Text Wrapping	v w
Horizontal Scrolling	v h

Standard JIRA Actions

Action	Shortcut
Operations Dialog	`.`
Edit Issue	`e`
Comment on Issue	`m`
Assign Issue	`a`
Assign Issue to Me	`i`
Edit Issue Labels	`l`
Actions Drop-Down	⌥↓

Changing Structure

Action	Shortcut
Move Up	^↑

Action	Shortcut
Move Down	^↓
Indent	^→
Outdent	^←
Drag and Drop	⇧Drag
Drag and Drop (Copy)	⌥⇧Drag
New Issue	⏎
New Sub-Issue	⇧⏎
Remove from Structure	⌦

Changing Issues

Action	Shortcut
Edit Field	*Double-Click*
Edit Summary	`tab`
Finish & Save	⏎ *or* ^⏎
Cancel Field Changes	`esc`
Edit Next Field	`tab` *or* ^⌥→
Edit Previous Field	⇧`tab` *or* ^⌥←

Action	Shortcut
Edit Next Issue	^⌥↓
Edit Previous Issue	^⌥↑

Selecting Issues

Action	Shortcut
Toggle Selection	space
Select All	⌘a
Select All Sub-Issues	⇧→
Deselect All Sub-Issues	⇧←
Expand Selection Down (Up)	⇧↓ (⇧↑)
Bulk Selection	⇧⇞ ⇧⇟ ⇧↖ ⇧↘
Cancel Selection	esc

Advanced

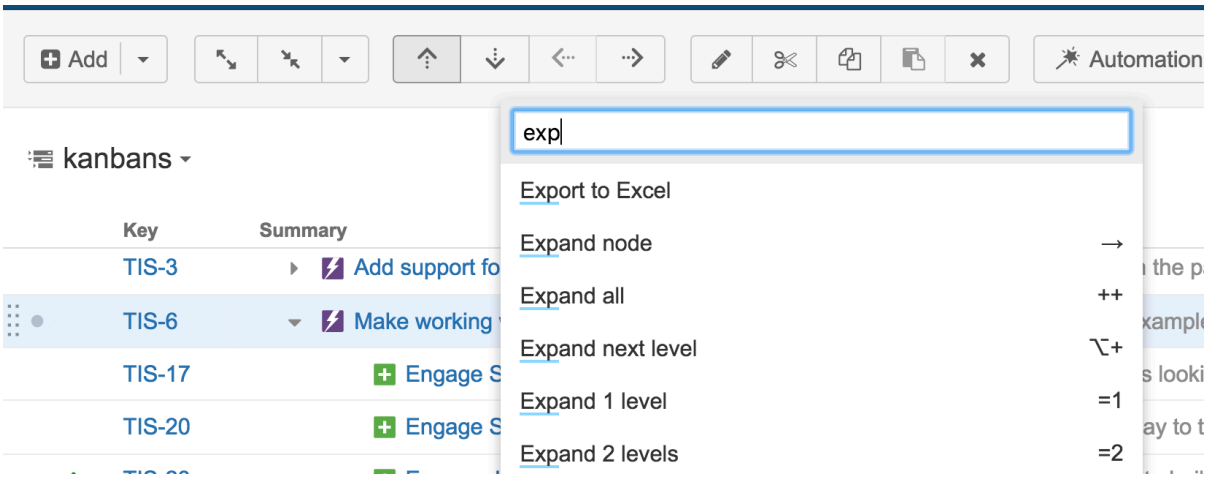
Action	Shortcut
Hide/Show Resolved	rr
Cut (Prepare to Move)	⌘x
Paste (Move)	⌘v

Action	Shortcut
Paste Sub-Issue (Move)	⌘⇧v
Fix/Unfix View on Issue	^.
View Full-Size Image(see page 467)	ii
Show/Hide Issue Details without Switching Panel	⇧o
Toggle Structure description	d d

3.2.7.3 Quick Action Lookup

If you're not sure how to do something in Structure, use the special keyboard combination **s,q** (press **s** and then press **q**).

This will pull up the "Action Lookup" input box, where you can type what you need to do and see a list of available "actions" that match your description.



Action Lookup also displays the keyboard shortcut associated with each action.

3.3 Working with Issues

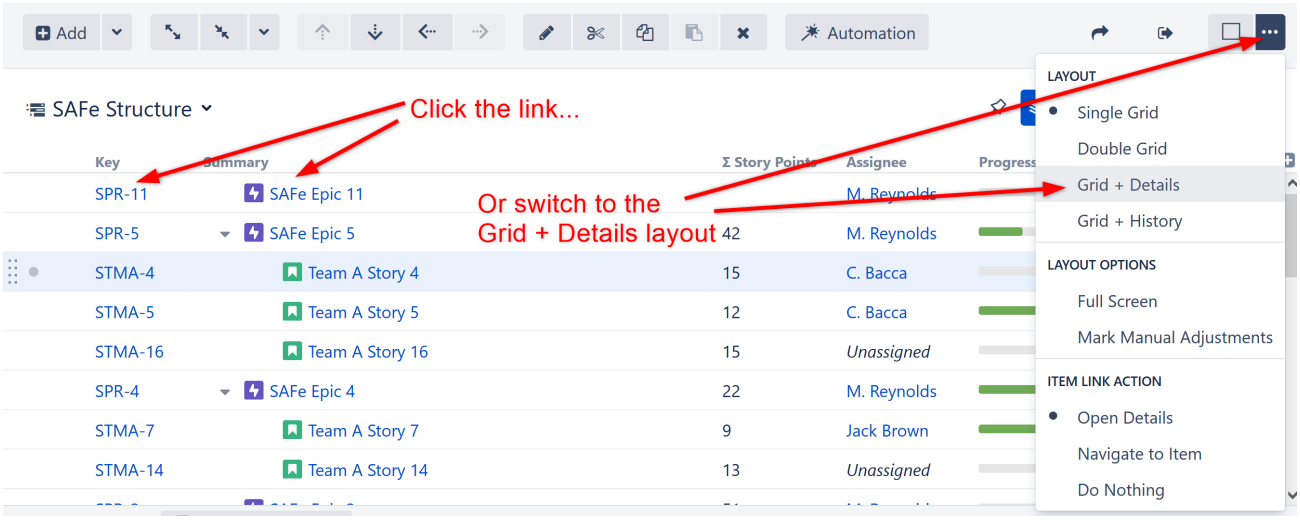
Select one of the following articles to see how you can work with issues from within a structure.

- Viewing Issue Details(see page 126)
- Creating New Issues(see page 128)

3.3.1 Viewing Issue Details

As you work with a structure or search results on the Structure Board, you can open the full issue information in the Issue Details Panel on the right.

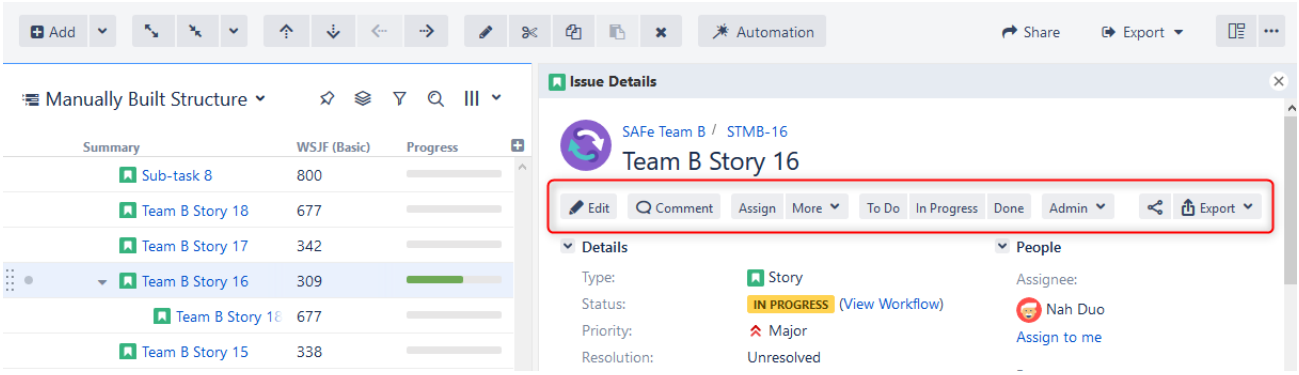
To open the Issue Details Panel, click the issue link (key or summary) or select it in the **Toggle Panels** menu:



> You can define what happens when you click the issue link in a structure. By default, the Issue Details panel is opened. It can be set to open the standard JIRA issue page instead, or do nothing.
>
> To change the default action, go to the **Toggle Panels** menu and select your preference in the **Item Link Action** section.

3.3.1.1 Working with an Issue

In the details panel, you can work with the issue in the same way you can within the Jira Issue Navigator: edit[22], view and add comments[23], share[24], view history[25], view development information[26], and more. For specific information on working with and editing issues, please refer to the Jira documentation[27].



To see details for another issue in the structure, simply select another issue(see page 82) by clicking it or moving to it with the arrow keys.

To close the issue details panel, click the close button in the top right corner.

> You can also open the issue page in a separate browser tab or window by pressing `Ctrl` (Mac: `Cmd`) or `Shift` while clicking the issue key or summary link.

3.3.1.2 Separate View for Issue Details

In order to provide enough room to view all the information in the Issue Details Panel, the Structure panel automatically switches to a compact view (with only *Key* and *Summary* columns visible) when the details panel is opened.

Structure will switch back to your previous view when the details panel is closed.

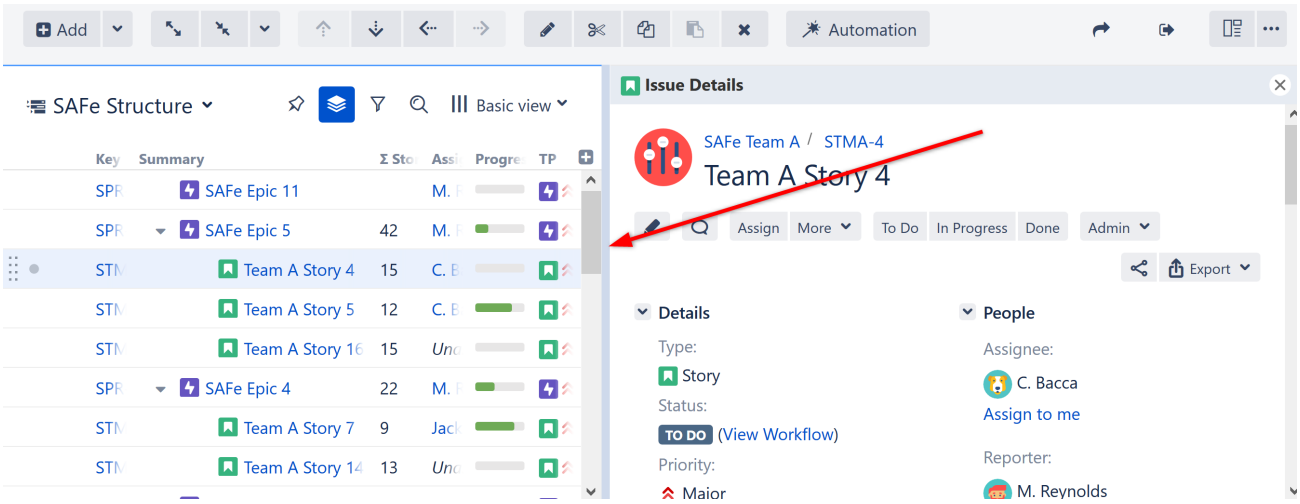
> You can adjust the default views in View Settings(see page 541).

3.3.1.3 Resizing the Issue Details Panel

You can divide the horizontal space between the details panel and the main panel by dragging the separating border. Structure remembers the ratio of the details panel width to the window width, and it will maintain that ratio when you open Structure Board again or resize your browser's window.

---

22 https://confluence.atlassian.com/display/JIRA/Editing+an+Issue
23 https://confluence.atlassian.com/display/JIRA/Commenting+on+an+Issue
24 https://confluence.atlassian.com/display/JIRA/Emailing+an+Issue
25 https://confluence.atlassian.com/display/JIRA/Viewing+an+Issue%27s+Change+History
26 https://confluence.atlassian.com/display/JIRA/Viewing+the+Code+Development+Information+for+an+Issue
27 https://confluence.atlassian.com/display/JIRA/Working+with+an+Issue



Details panel width is remembered for the selected view. Thus, if you select another view and adjust the details panel width, the original width will be restored when you select the original view.

3.3.1.4 Details and Secondary Panels

If you have the secondary panel (see page 518)open, when you click an issue in the main panel, the secondary panel will be hidden while the details panel is open. To restore the Secondary Panel, close the Issue Details Panel.

3.3.1.5 Using the Keyboard

Use **o** or **Shift+o** to show/hide the details panel.

As with the secondary panel, you can switch keyboard focus between panels using the **** (backslash) shortcut. When the focus is in the Issue Details Panel, keys like ↑ or ↓ (also `PgUp`, `PgDn`, `Home`, `End`, `or ↟, ↡, ↖, ↘`) scroll the details panel, while all other Structure shortcuts(see page 116) work as usual (including `j` and `k`, which select the next/previous issue in the structure). All shortcuts available to you on the Issue Page should also work as usual: **,** (comma) should open the field selector, **e** should open the Edit Issue dialog, etc.

When you open the details panel with **o**, the details panel is automatically focused. **Shift+o** does not switch focus.

3.3.2 Creating New Issues

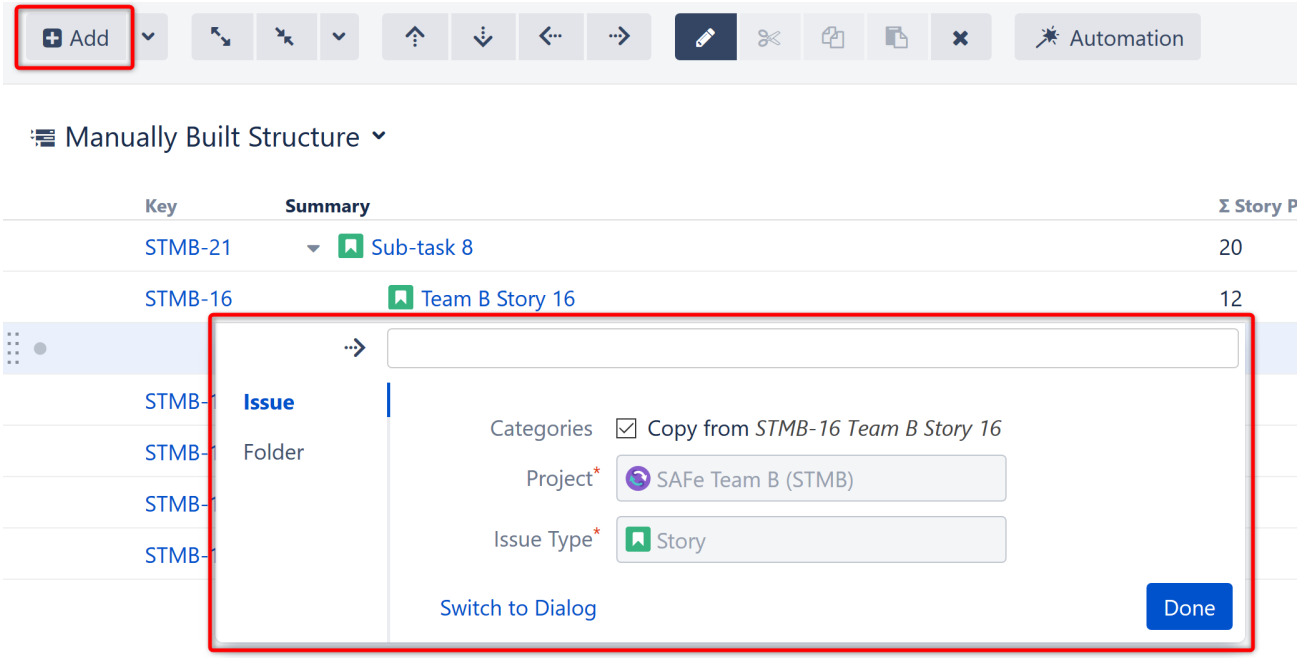
You can quickly create new issues and folders right in Structure, or you can use the standard "Create Issue" dialog and have new issues added to your structure automatically.

3.3.2.1 Create a New Issue in Structure

To create a new issue beneath the currently selected item:

1. Use the **Add** button in the toolbar.
2. Enter the Issue Summary in the top entry box of the New Issue panel (to the right of the arrow).

3. To copy attributes (project, issue type, assignee, version information, etc.) from the last selected issue, make sure the **Categories** checkbox is selected. To enter all new information, uncheck this box.
4. Press **Enter** or click **Done** to finish editing and create the new issue on the server.



Keyboard Shortcut

Save some time! To create a new issue, use the keyboard shortcut: **Enter**.

To create a new sub-issue of the currently selected item, use the shortcut: **Shift+Enter**.

Press **Escape** to cancel creating a new issue.

Categories: Copy from...

When you choose the "Copy from..." option, the following fields are copied from the last selected issue:

- Project and Issue Type
- Parent Issue, if the cloned issue is a JIRA sub-task
- Component, Affects Versions, Fix Versions, Environment, Assignee, Priority, Security Level
- All custom fields that are **required** by the fields configuration for that particular Project and Issue Type

Please note that the archived versions are skipped when copying Affects Versions, Fix Versions and version-based custom fields.

Editing Other Fields during Creation

The New Issue panel only allows you to select the project and the issue type (or copy some fields from another issue). If you need to edit other fields:

1. Before you start creating the new issue, add the columns you need to edit to your current view
2. Click **Add** to open the New Issue panel

3. To get to the fields you need to edit, hold **Ctrl+Alt** while using the arrow keys to move between the available columns
4. Once you have finished editing the columns, hold Ctrl+Alt and press the Down arrow to return to the New Issue pane.

You can also edit other fields by clicking the **Switch to Dialog** button, which opens the "Create Issue" dialog(see page 130).

> ⚠ If you have any required fields, you must enter a value into those fields (or have default values defined). Otherwise, you will be unable to create the new issue.
>
> To correct this, follow the steps above and make sure all required fields are filled in.

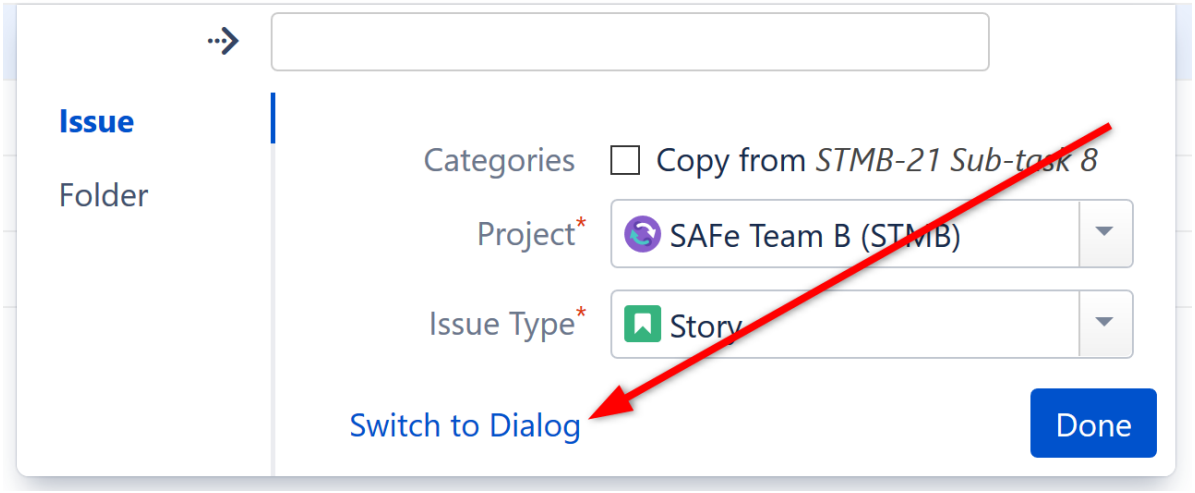
Creating Sub-Tasks

It is not possible to create a Jira sub-task from scratch in Structure. However, if you have an existing sub-task in your structure, you can create another sub-task with the same parent using the **Copy from..** option.

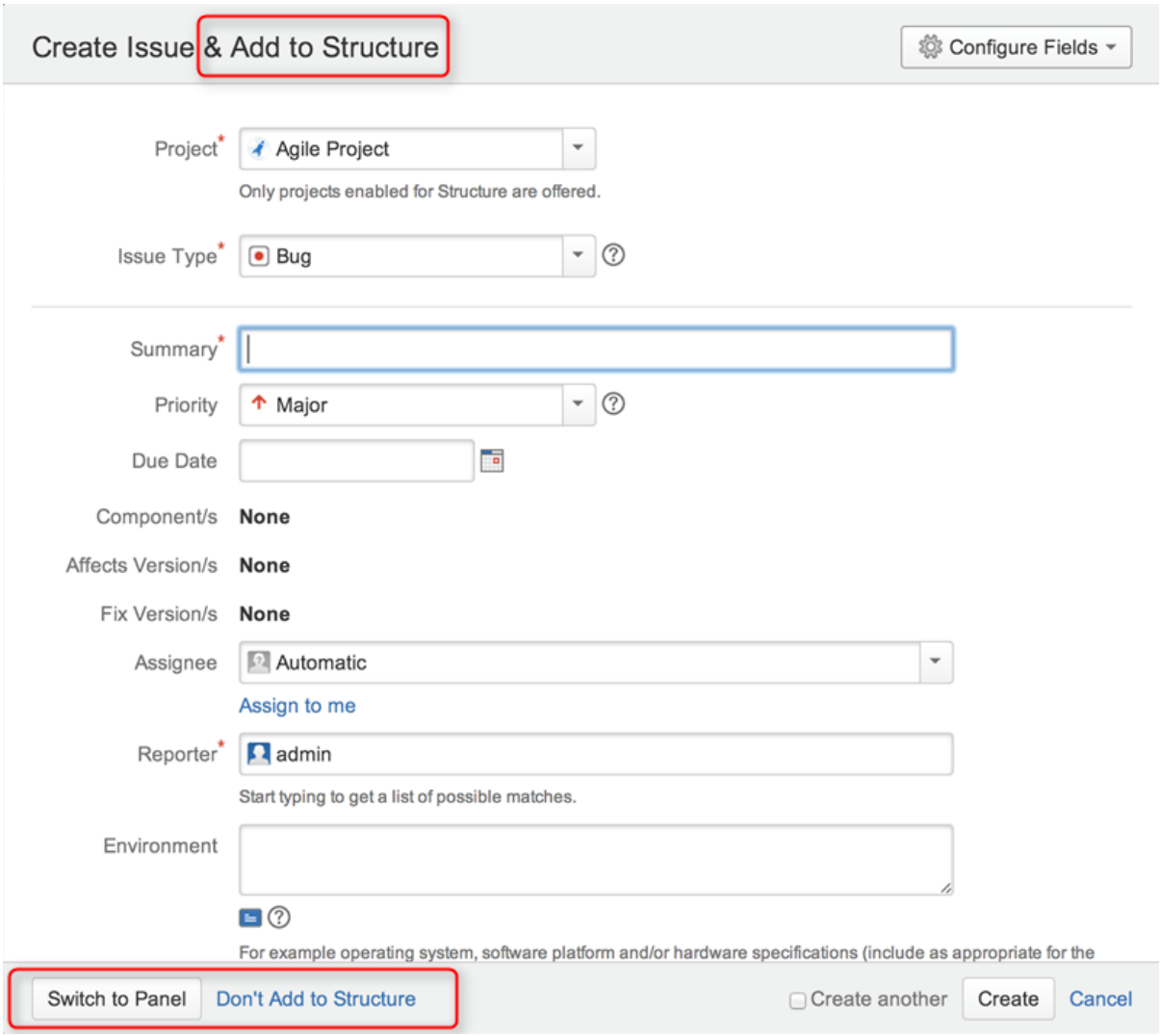
> ⓘ This refers to the Jira sub-task issue type, not Structure sub-issues.

3.3.2.2 Create Issues Using the "Create Issue" Dialog

If you need to update fields not currently in your view or that are not editable, click the **Switch To Dialog** at the bottom of the New Issue panel.



This will switch you to Jira's **Create Issue** dialog, which allows you to enter information into fields just as you can when creating a new issue directly from Jira. Once you finish entering information and click **Create**, the new issue is automatically added to the structure.



If you don't want the new issue added to the current structure, click **Don't Add to Structure** at the bottom of the dialog.

> To switch back to the Structure New Issue panel, click **Switch to Panel.** This will preserve all entered data and populate existing columns if possible. You can also switch back to dialog mode at any time. The system will remember the last-used mode (dialog or panel) and use it the next time you start creating a new issue.

3.3.2.3 Creating Epics

When creating Epics, the Epic Name custom field is required. To simplify the process of creating multiple epics, Structure will copy the new epic's summary to its Epic Name field, if the latter is empty. This way you can simply type an epic name into the Summary field, and proceed to the next issue.

The copying only happens once, when an epic is created. You can change the summary or the epic name at a later time, if you want them to be distinct. Alternatively, you can also add the Epic Name column to the table and enter new epic names explicitly.

3.3.2.4 Additional Keyboard Shortcuts

Immediately after you have press **Enter** or **Shift+Enter** or **Insert** to start editing a new issue, you can also use the keyboard to change the creation mode.

Use the following keyboard shortcuts **while the summary field is still empty**:

Enter or **Tab**	Cycle through Project, Issue Type and Summary field. When Project or Type field is selected, use arrows or start typing to select a project or type.
Ctrl+Enter / **Cmd+Enter**	Toggles cloning mode (**Categories: Copy** checkbox).
Alt+Enter / **Option+Enter**	Switches editor to dialog mode and back to panel.

If you have already entered the summary, you can use the mouse to change the creation mode, project or issue type.

3.3.2.5 Uploading New Issue to the Server

After you create a new issue, Structure only displays the Summary field until it receives confirmation from Jira that the issue has been created on the server. Once confirmation is received, the remaining columns for the issue will be loaded.

While the new issue is being uploaded to the server, you can start creating the next issue.

3.3.3 Editing Issues from Within Structure

Many issue fields can be edited directly from a structure.

To edit a value displayed in the Structure widget, do one of the following:

- **Double-click** that value
- Select the issue and click the **Edit** button  on the toolbar
- Select the issue and use a keyboard shortcut – **Tab** or **F2**

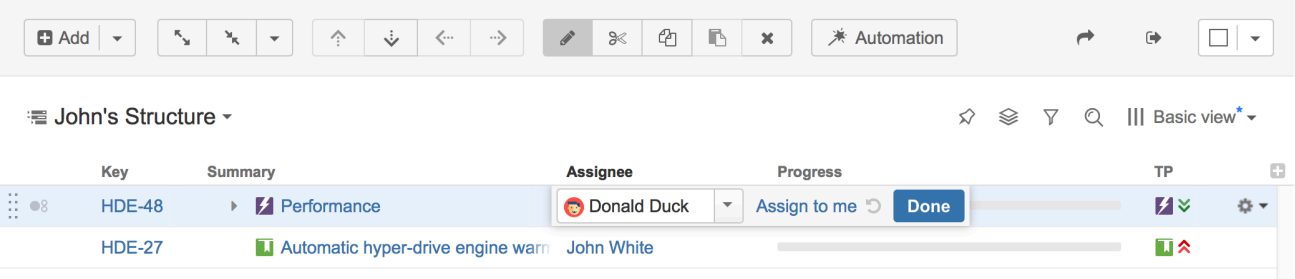
If the value is a link (like in the Summary or Assignee fields), you can still double-click it – the browser will not open the link and will start editing instead.

3.3.3.1 Edit Mode

Once you have entered the Edit Mode, you can edit one or more issues simply by clicking the values you need to edit, or navigating to them with special keyboard shortcuts (see Using Keyboard in Edit Mode(see page 132)).

When working in Edit Mode:

- A field editor appears over the cell you are editing
- The active column's name appears in bold in the table header
- The **Edit** button on the toolbar is toggled on



Using the field editor, enter the new information you want in each field. Once you have finished editing a value (or multiple values), click **Done**.

3.3.3.2 Learn More

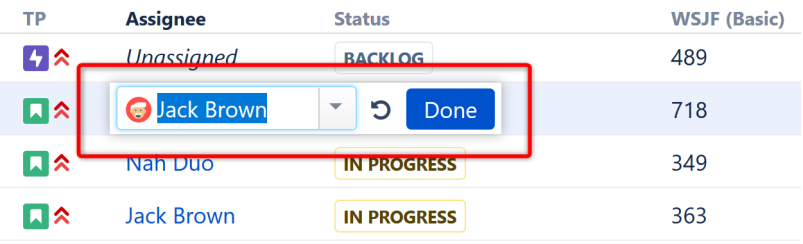
Editing works on every page where Structure widget is displayed; however, there are some limitations when editing issues from the Structure Gadget(see page 137).

- Changing Fields(see page 133)
- Keyboard Shortcuts in Edit Mode(see page 135)
- Correcting Input Errors(see page 137)
- Editing from Gadget(see page 137)
- On E-mail Notifications(see page 138)

> ⚠ In order to edit an issue's fields, you need Edit Issue permission for that issue. If you do not have the correct permission, a read-only flag(see page 461) will be displayed at the far left of the item row.

3.3.3.3 Changing Fields

When editing a field, make the change with the field editor and click **Done** (or hit **Enter**) to have the change saved on the server.



If you need to change more than one field, you can navigate through fields with the mouse or by clicking **Tab**, **Shift+Tab**, or **Ctrl+Alt+arrow**.

Your changes will automatically be saved to the server when you:

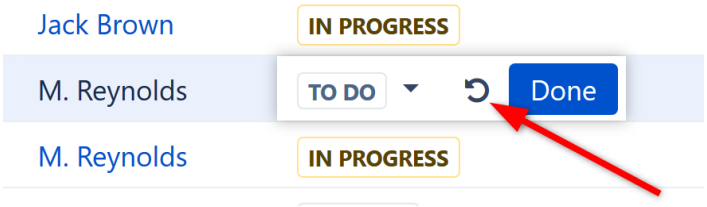
- Click Done
- Click the Edit icon in the toolbar to exit Edit Mode
- Switch to editing another issue, or
- Collapse the hierarchy

> If your Jira is configured to send e-mail notifications about changes, then a notification will be sent as soon as you have finished editing an issue - see On E-mail Notifications(see page 138).

Undoing Changes

To cancel changes and exit Edit Mode, click **Escape**.

To restore the original value of a field and stay in Edit Mode, click the **Revert Changes** button:

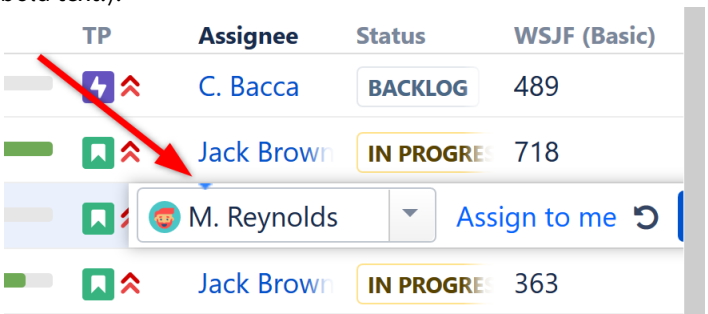


> Hitting **Escape** only reverts the value of the currently edited field. Any other changes will remain.
>
> For example, if you edit an issue's Summary, Assignee and Components fields, but you hit Escape while editing Components, the changes to Summary and Assignee will still be uploaded.

The Field Editor

The editor for each field is the same as the one used on the Edit Issue page, only a bit more compact:

- All help texts, descriptions and field labels are hidden. If you need to see help or the field description, hover the mouse pointer over the input field.
- Normally, the editor is aligned with the top left corner of the edited cell. However, if it does not fit horizontally on the page, its position is adjusted and a small blue triangle is shown to mark the place where the edited cell starts. You can also look at the table header to see which field is being edited (the one with bold text!).



Allowed Changes

In the Edit Mode, you can change fields that are added to the Edit Screen for the edited issue. If a field is not on the Edit Screen, or if it can't be edited directly (such as the Resolution field), the editor won't be shown, or it will display a corresponding error.

Additionally, some fields may have unique limitations. For example:

- The Status field can only be edited if there are no required fields or screens on transition between statuses
- The Original Estimate field is not editable after work has been logged (in Jira's legacy time tracking mode)

3.3.3.4 Keyboard Shortcuts in Edit Mode

You can use keyboard shortcuts to quickly edit issues within Structure.

Entering Edit Mode

Keyboard Shortcut	**Action**
Tab *or* **F2**	Enters Edit Mode for the currently selected issue, starting with the Summary field or the last-edited field.
Enter **Insert** or **Shift+Enter**	Enters Edit Mode for a new issue (Enter) or sub-issue (Insert/ Shift+Enter).

Keyboard Shortcuts in the Edit Mode

Keyboard Shortcut	**Action**
Enter **Ctrl+Enter** *(in large text fields)*	Exit Edit Mode and save all values on the server.
Escape *(hit twice in combo boxes and drop-downs)*	Revert the current field to the value that was there before editing started and exit Edit Mode. *Note: Pending changes in other fields will be saved on the server.*
Tab	Edit next editable field. If the current field is the last editable field for the selected issue, start editing next issue.
Shift+Tab	Edit previous editable field. If the current field is the first editable field for the selected issue, start editing previous issue.

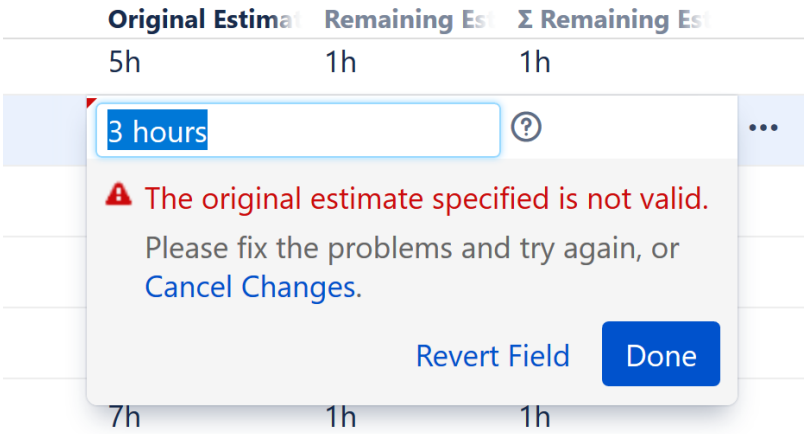
Keyboard Shortcut	Action
Ctrl+Alt+↓	Edit the same field of the next editable issue.
Ctrl+Alt+↑	Edit the same field of the previous editable issue.
Ctrl+Alt+→	Edit next editable field. Unlike **Tab**, this combination will not move editing to the next issue.
Ctrl+Alt+←	Edit previous editable field. Unlike **Shift+Tab**, this combination will not move editing to the previous issue.
↓ *or* **Alt+↓** *(in drop-downs)*	Opens the drop-down list or selects the next value in the list. If the drop-down is shown, use **Enter** to select a value or **Escape** to cancel selection.
Alt+↓ *(in date/time fields)*	Opens the date picker. Use arrows to navigate dates in the date picker; use **Enter** to select a date or **Escape** to close the date picker.
→ and ←	Move between multiple fields on the same editor (for example, between the two editors of a Cascade custom field). Does not work if the input is a text field.
↓ and **↑** *(for checkboxes and radio buttons)*	Move between multiple fields on the same editor (for example, between the checkboxes of a Multiple Checkboxes custom field).
Space	Select / unselect a checkbox or a radio button.
↓, **↑**, **Shift+↓**, **Shift+↑**	Select / unselect values in a Multi-Select custom field.

Note that the **Tab** key moves editing to the next cell, so if you have multiple input fields on a single field editor, you need to use arrow keys to switch between them.

See Also: Keyboard Shortcuts(see page 116)

3.3.3.5 Correcting Input Errors

If you enter an incorrect value when editing a field, or if there are any other problems saving that value on the server, Structure will display a warning message and mark the cells with problems.



Click the warning message or the cell with the error to enter Edit Mode, see problem details and correct the error. You can:

- correct the value and hit **Enter** or click **Done** to try to save the values on the server again
- click **Revert Field** to restore a previous value of the field, known to be valid, or
- click **Cancel Changes** to cancel all changes to this issue, including possible changes to other fields.

You can edit other issues and continue working with Structure before fixing the editing problem. However, it is advised to correct the error as soon as possible.

Input Errors when Creating a New Issue

If the error happens when saving a new issue on the server, saving any further changes on the server is suspended – until the error is fixed or the creation of the new issue is cancelled. This is a necessary measure, because the success of the following changes may depend on the successful creation of that new issue.

When you have errors in the fields of a new issue, fix them as soon as possible or cancel the creation of that issue. Otherwise, any further changes are not uploaded until the problem is fixed – and you risk losing them!

To cancel the creation of a new issue, select it and click the **Delete** button or press the **Delete** key.

3.3.3.6 Editing from Gadget

The Structure Dashboard Gadget allows editing issues too, but due to some incompatibilities between field editors and gadget framework, not all fields can be edited.

Structure Notes(see page 470) and the following Jira fields are editable from Structure Gadget:

- Summary
- Assignee
- Issue Type
- Priority
- Reporter
- Security Level
- Original Estimate
- Remaining Estimate
- Numeric Custom fields (like Storypoints)
- Text Field (multi-line)
- Single and Multi Select Fields from Custom Fields ++[28]

To edit other fields, open Structure Board, an issue page or any other page with the structure.

To be able to edit the fields in the gadget, the user should have permissions to edit them, and the fields should be present on the Edit Screen.

The Confluence gadget is not editable.

3.3.3.7 On E-mail Notifications

When issues are updated in Structure, notifications are sent based on the corresponding project's notification settings in Jira. This usually means everyone involved with the issue will receive an email notification.

When editing an issue within Structure, emails are sent only once the changes are sent to the server. This happens when you:

- Hit the **Done** button or **Enter** key, or
- Start editing another issue.

To minimize the number of email notifications sent, we suggest following these steps when editing issues in Structure:

- **Editing a single issue** - If you need to change several fields of an issue, edit one field and then navigate to the next field. Only hit **Done** or **Enter** when you have finished editing all fields.
- **Editing multiple issues** - Update all the fields for a single issue before moving to the next.

Jira admins and project admins can turn off notifications for specific groups of issues using the Jira bulk operations wizard - see Bulk Change(see page 138).

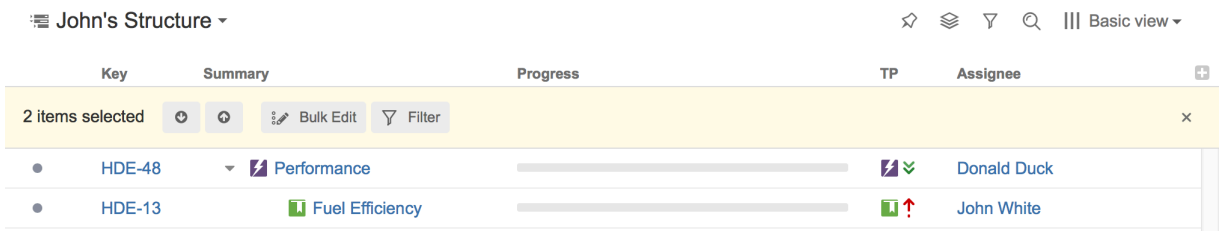
3.3.4 Bulk Change

With Structure, you can quickly select multiple issues(see page 113) and edit them using the Jira bulk operation wizard.

28 https://marketplace.atlassian.com/apps/1216682/custom-fields-for-jira?hosting=server&tab=overview

To edit multiple issues:

1. Select the issues by clicking on their issue selectors (the gray dot at the left of each item row) or pressing *Space*, *Shift+Space* or other Keyboard Shortcuts(see page 116) for selecting issues.
2. Click **Bulk Edit**.



This will open the Jira bulk operation wizard, which allows you to perform bulk changes on all the selected issues. Once you finish making changes, the browser will be redirected back to Structure Board.

3.3.5 Cloning Multiple Issues

Structure allows you to copy the whole structure and clone all issues in the structure. See Copying Structure and Cloning Issues(see page 548).

If you need to clone only some of the issue in the structure, you can use the following procedure:

1. Select issues you'd like to clone using multiple selection(see page 113).
2. Use **Copy** action on the toolbar (or hit Ctrl+C / Command+C) to copy the issues to the Issue Clipboard(see page 111).
3. Use **Structure | Create Structure** menu and create a new temporary structure, let's call it **T1**.
4. Open the new structure and use **Paste** action to add issues from clipboard.
5. Copy and clone structure **T1** – see Copying Structure and Cloning Issues(see page 548). Let's name the resulting copy **T2**.
6. Open **T2**, select all issues (use Ctrl+A / Command+A).
7. Use issue clipboard in the same way to copy cloned issues back to the structure where they are needed.
8. Delete structures **T1** and **T2**.

3.3.6 Using Jira Actions

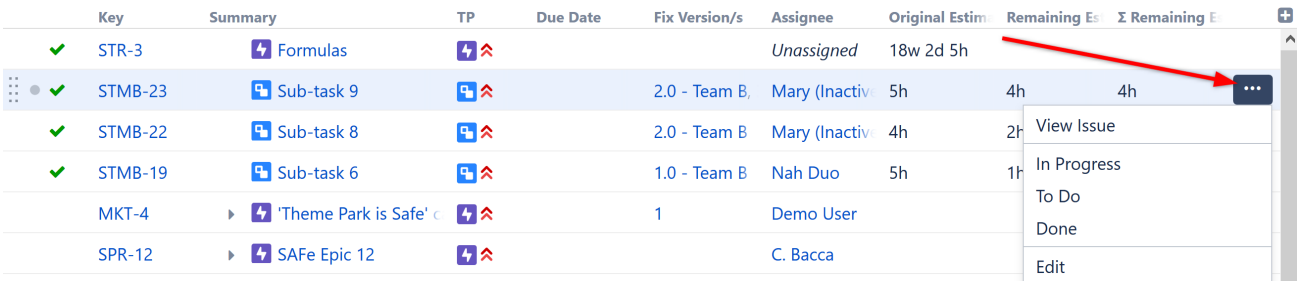
Jira actions can be applied to issues directly from Structure, using the Jira Actions drop-down and Jira keyboard shortcuts.

3.3.6.1 Using Actions Drop-Down

The Jira Actions drop-down includes the most frequently used actions and operations available for the selected issue.

To use an action:

1. Click on the **three dots** at the right side of the issue's row (it will not be visible until you mouse over it), or select the issue with the keyboard and hit **Alt+Down Arrow**.
2. Select the action with the mouse, or use the **Up/Down Arrow** keys and **Enter** to select the action with the keyboard.



The Jira Actions menu is unavailable when the Issue Details panel (see page 126)is opened.

3.3.6.2 Using Jira Shortcuts

Most Jira shortcuts that are available on the Issue Navigator page also work in Structure. Just select an issue and hit the shortcut.

The most useful shortcut is "." (dot) - available since Jira 4.2 - which lets you type in the name of the action you need performed.

Calling an action usually brings up a dialog or moves the browser to another page. Please pay attention to the dialog title or the window title to see that you're applying the action to the correct issue.

On the Issue Page(see page 90), keyboard shortcuts are always applied to the viewed issue - regardless of the selection in the structure.

3.3.6.3 No Page Reload

In many cases, Structure is able to proceed without a page reload after you have applied a Jira action to an issue. The applied changes are immediately visible in the structure, providing a very smooth experience when working with a collection of issues.

Whether a page is reloaded after an action is applied depends on which page you are using to work with issues, and what action is being applied. On the Structure Board(see page 88), most actions do not require page a reload.

3.4 Generators

Generators (Automation) are powerful tools that let you create **dynamic structures**, which will update themselves when there are changes in Jira (and can update Jira when you make changes in the structure).

You can make parts of a manually created structure dynamic – for example, automatically place all issues that match a query under a manually added folder – or you can build your entire structure using automation.

Sorry, the widget is not supported in this export.
But you can reach it using the following URL:

https://www.youtube.com/watch?v=Q343-zviSX4

3.4.1 How Automation Works

Automation works through **generators** – special rules that tell Structure what issues to show you from Jira and where to place them within the structure. We like to think of this as the "skeleton" of a structure.

Every time you open a structure, these generators will run again and completely rebuild the structure, based on the current information available in Jira. In fact, Structure will continue to check for changes even while the structure is open, ensuring that the information you are seeing is up-to-date, without needing to reload the page.

3.4.2 Types of Generators

- Insert(see page 142) - Automatically add issues to the structure.
- Extend(see page 148) - Add additional issues based on issues already in the structure.
- Filter(see page 158) - Remove issues that do not pass certain criteria.
- Group(see page 169) - Group issues by most standard Jira fields, custom fields, issue attributes, and issue links.
- Sort(see page 140) - Order the structure based on Jira fields.

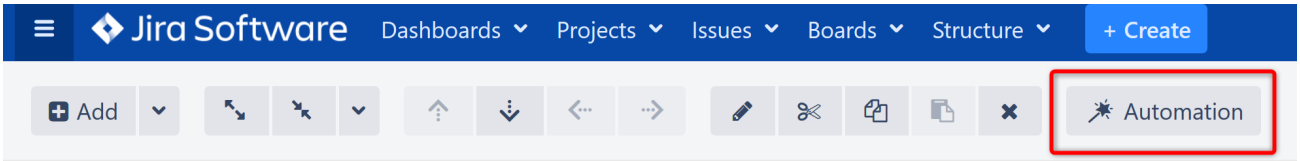
3.4.3 Generator Scope

Generators are added right inside the structure, just like other items, and their scope is defined by their position within the structure. Place the generator at the top of the structure, and it will impact the entire structure. Place it under a folder, and it will only affect items within that folder.

3.4.4 Adding a Generator

3.4.4.1 Automation Editing Mode

Before you can add a generator to your structure, you need to enter Automation Editing Mode. To do this, simply click the **Automation** button in the Structure toolbar.



As you click the button, the name of the structure will appear at the very top (**Root**) of the structure - this item works as a parent for the whole structure.

If the structure already contains generators, you will see those displayed in the structure as well.

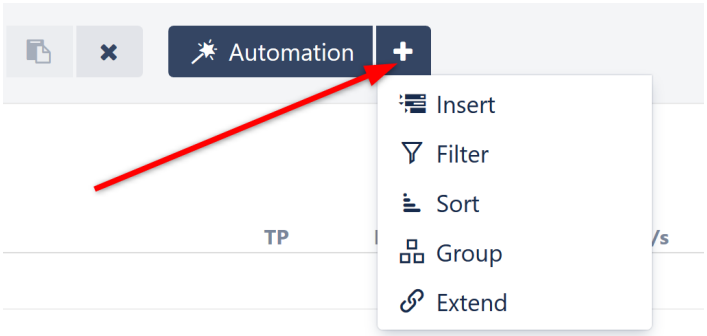
3.4.4.2 Adding a Generator

Before adding a generator, you must select where you want to place it within the structure. A generator's placement defines its scope:

- If you want the generator to affect the entire structure, place it at the top of the structure by selecting the structure name in the top row.
- If you want the generator to only affect part of the structure, select a static item, under which the generator will be applied.

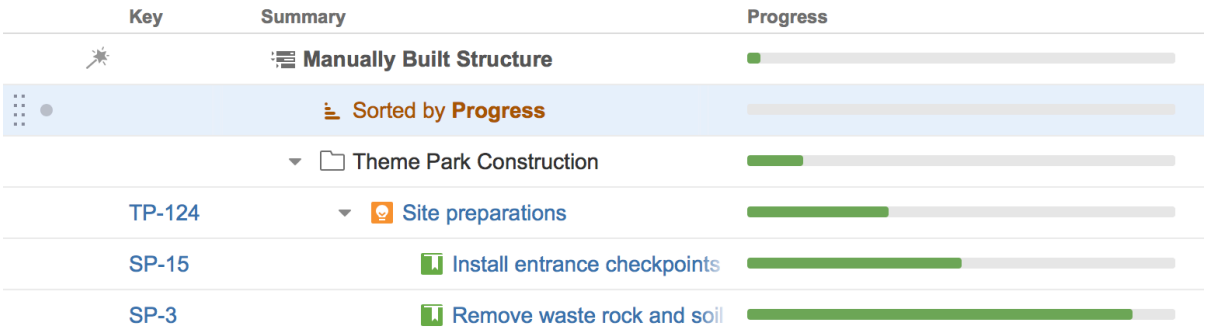
> You can only add generators under static parts of the structure. You cannot add them under dynamic items (items added by other generators).

Next, click the '**+**' icon next to the Automation button and select the type of generator you want to add.



Each option will allow you to configure specific rules, which will be used to determine which issues appear in the structure. For more information about each generator and their options, see Types of Generators[29].

Once you have entered the appropriate rules for your structure, click **Apply** to add the generator to your structure.



To hide the list of generators, click the **Automation** button again.

3.4.5 Insert Generators

Insert generators allow you to automatically add issues to the structure.

29 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4

This is often the first step in building a new structure using automation, since it allows you to import specific issues that you can then sort, group, filter or expand upon using additional generators.

Some examples of how you might use the Insert generator include:

- Add all Epics from a specific Agile board (and then you could use the Extend Generator(see page 148) to add stories and sub-tasks)
- Use a JQL query to add all issues assigned to you (and then organize them with a Group Generator(see page 169) or Sort Generator(see page 142))
- Insert issues from specific projects into their own folders within the structure
- Compile multiple structures into a single structure

Insert generators can only add issues from Structure-enabled projects(see page 954).

3.4.5.1 Types of Insert Generators

As you add an Insert generator, you can choose from the following options:

- JQL Query Inserter(see page 143) — Adds the results of the JQL query to the structure.
- Agile Board Insert(see page 145) — Adds issues from a selected Agile board into the structure.
- Structure Inserter(see page 146) — Adds a selected structure into the current structure.
- Text Query Inserter(see page 147) — Adds the results of a text search to the structure.

Additional Insert generators may also be available, depending on the add-ons you have installed. For instance, if you have Structure.Pages installed, you will see a Confluence Pages Inserter in your drop-down list.

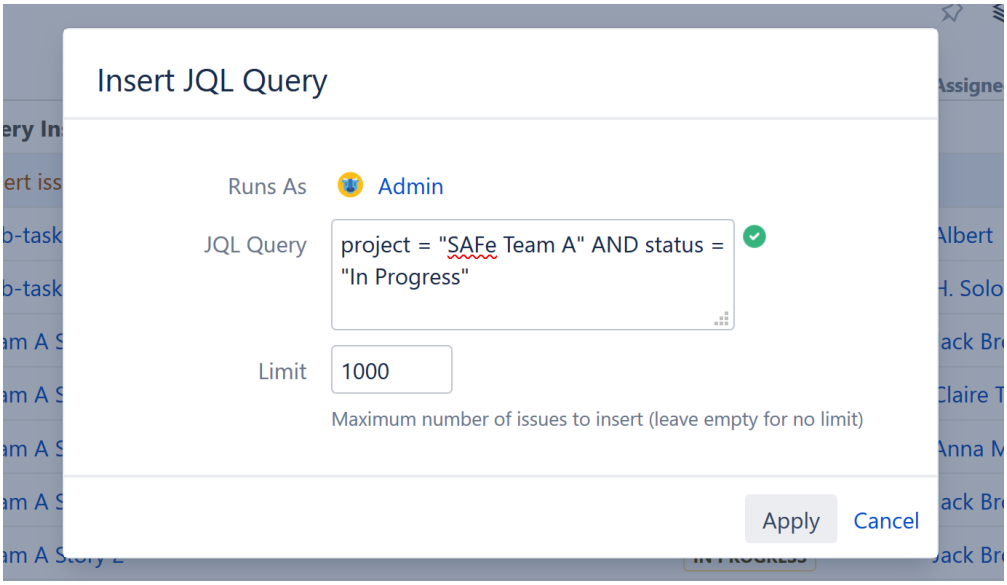
3.4.5.2 Always Up to Date

Generators run every time you open a structure, so the list of issues added by the Insert generator is always up to date.

Additionally, if issues change as you work with the structure, they will be added or removed accordingly, based on the rules of your Insert generator.

3.4.5.3 JQL Query Inserter

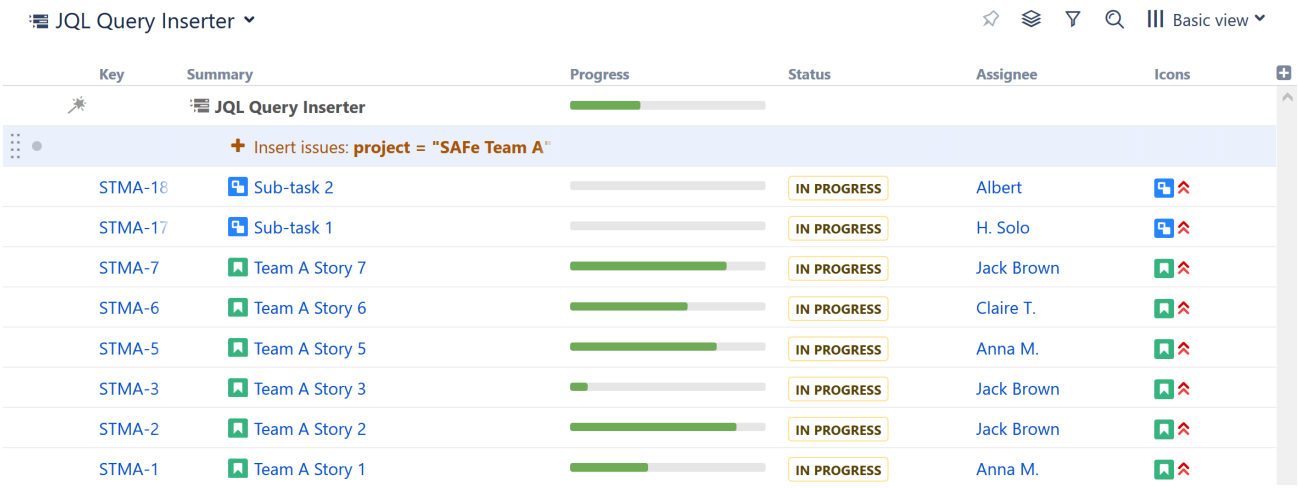
The JQL Query Inserter allows you to add issues based on a JQL query.



To determine the scope of issues that should be added to the structure, simply add the appropriate JQL query[30].

Because the JQL Inserter allows you to include issues from multiple projects (or even every structure-enabled project you have access to), it may result in very large structures. To limit the number of issues the inserter adds, enter an appropriate number in the **Limit** field. By default, the limit is set at 1,000 issues. If you don't want to limit the number of issues, simply leave this field blank.

Once you've entered your query and set your Limit, click **Apply**. The inserter should now appear as a new row in your structure, with the added issues placed below it.



When to Use the JQL Query Inserter

The JQL Inserter is a very powerful generator, because it allows you to set specific conditions, such as which projects should be included, what issue types, who issues are assigned to and more.

Here are a few examples of when you might want to use the JQL Inserter:

- You need to include issues from multiple projects (or even every structure-enabled project you have access to)

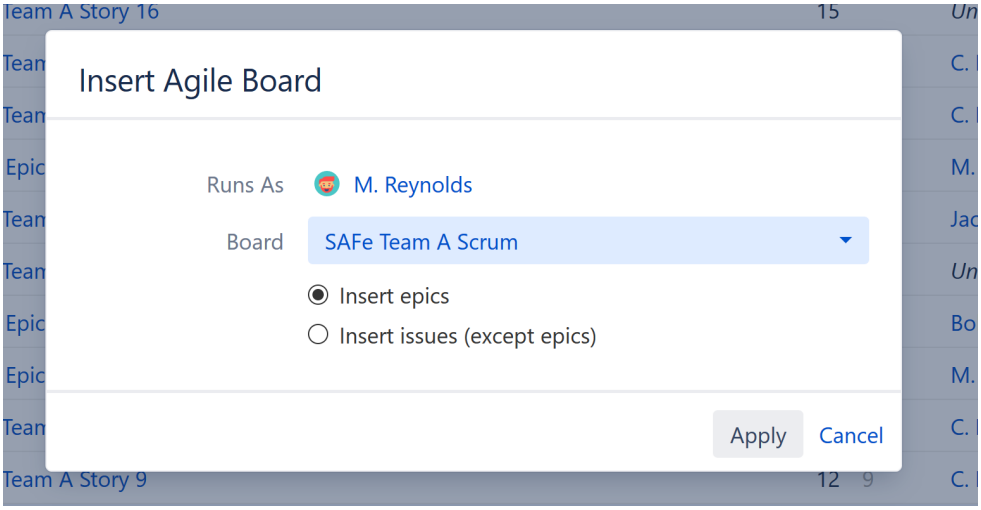
30 https://confluence.atlassian.com/jiracoreserver073/advanced-searching-861257209.html

- You need to add issues based on a specific Jira field. For example, you may want to add every issue assigned to a specific team, add every bug, only include issues from certain versions, etc.
- You need to narrow the scope of issues by including multiple parameters. For example, you could add epics assigned to the current user, open issues from a specific project or tasks that are overdue.

The JQL Query Inserter is a versatile tool, but it doesn't have to do all the work. Inserters are most effective when coupled with Extend(see page 148), Filter(see page 158), Sort(see page 143) and Group(see page 169) generators.

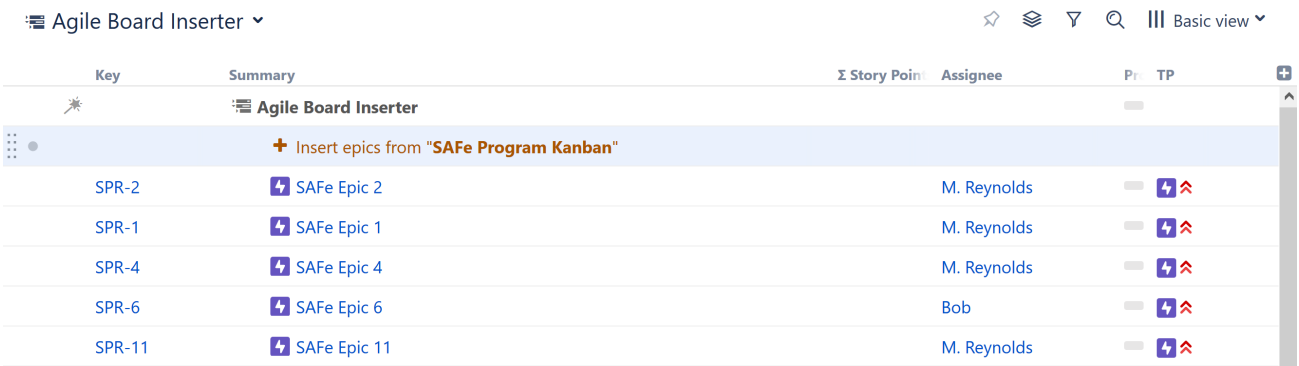
3.4.5.4 Agile Board Insert

The Agile Board Insert generator allows you to add issues (epics or all other issue types) from an agile board.



You can select from any Agile Boards that you have access to. Once you've chosen the Agile Board, select whether you want to insert epics or all other issues (stories, bugs, tasks, etc). Don't worry if you want to include both – select one for now, and you can add the other using Extend(see page 148) or Group(see page 169) generators later.

Once you've made your selection, click **Apply**. The inserter should now appear as a new row in your structure, with the added epics or issues placed below it.



If you want to include items from more than one Agile Board, you can add an additional Agile Board Inserter – or use the JQL Inserter,(see page 143) which allows you to specify multiple projects using the 'OR' keyword.

When to Use the Agile Board Inserter

The Agile Board Inserter is a particularly useful way to start a more complex structure.

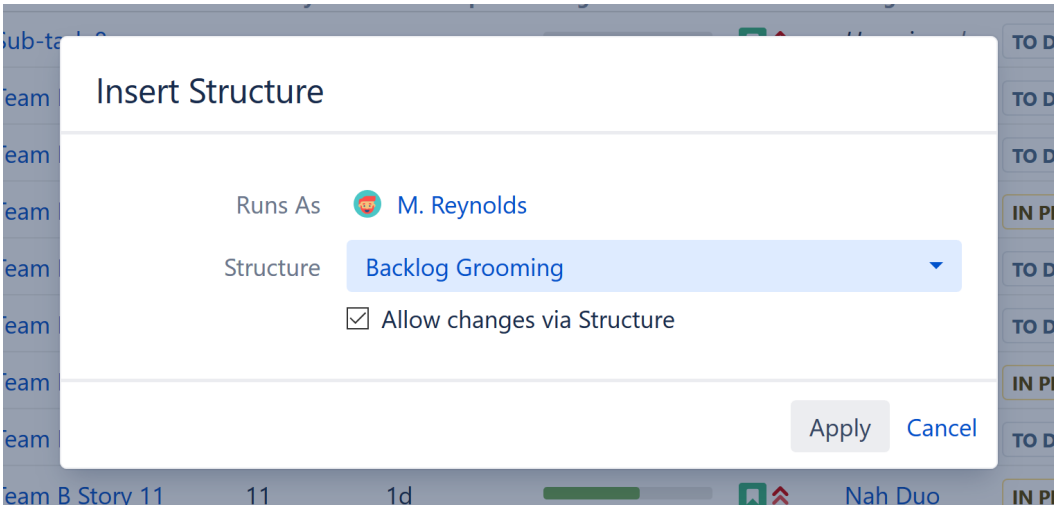
Here's a quick example of what that might look like:

1. First, add all the epics from a specific Agile Board.
2. Next, use the Extend (see page 148)generator to add stories beneath each epic.
3. Then use another Extend generator to add linked issues under those stories. The resulting hierarchy will allow you to quickly review stories and their dependencies.
4. Finally, you can add Sort (see page 145)or Filter(see page 158) generators to tailor the results to your specific needs – or use Transformations(see page 215) to temporarily sort or filter your results.

You can expand and customize Inserter results using the Extend(see page 148), Filter(see page 158), Sort(see page 145) and/or Group(see page 169) generators.

3.4.5.5 Structure Inserter

The Structure Inserter makes it possible to add an existing structure into the current structure. This can be extremely useful if you need to view multiple structures at a single glance.



You can insert any structure that you have access to.

You can only add one structure per inserter. If you need to include additional structures, simply add more inserters. You can add as many as you need!

We recommend that you create a special folder for each structure you want to add, and place the Structure Inserter inside that folder. This way, the inserted structure will be contained within the folder – so you can easily see where it begins and ends. If you are adding more than one structure, it's best to place them each in their own folder.

Allow Changes Via Structure

When the **Allow changes via Structure** box is checked, you can update the inserted structure right from your new structure. Any changes you make from the new structure will be reflected in the original

If you only want to view the contents of the original structure, uncheck the **Allow changes via Structure** box. The added structure will be read-only, so there's no risk of accidentally changing it.

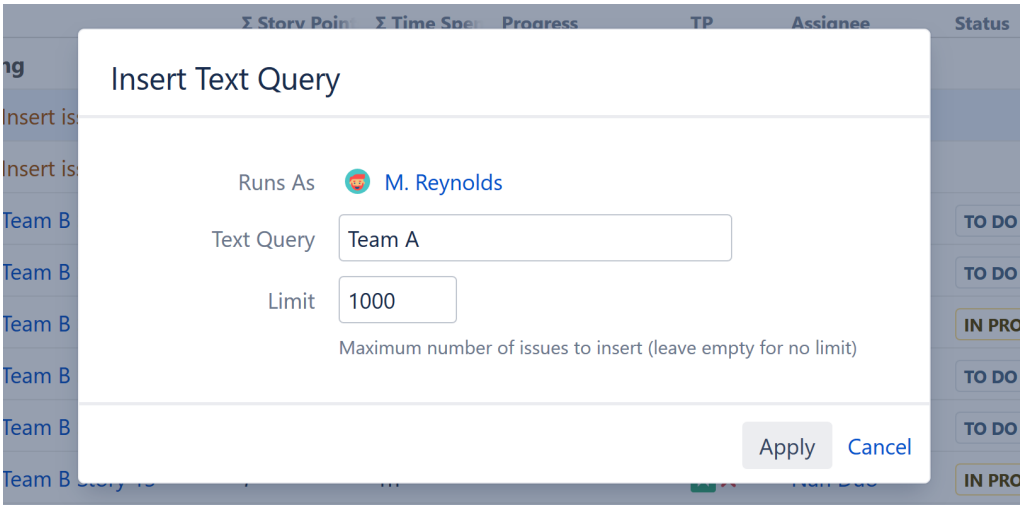
When to Use the Structure Inserter

The Structure Inserter is a great way to compile multiple structures for simplified viewing. This is particularly useful if you're responsible for multiple structures and need to quickly review them together.

For example, if several teams work with their own structures, you may want to create a structure with an overview of them all.

3.4.5.6 Text Query Inserter

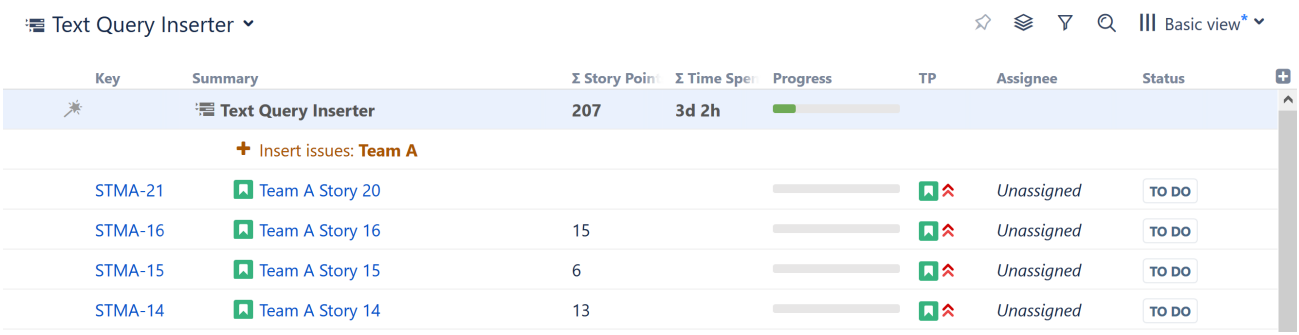
The Text Query Inserter allows you to add issues based on the text contained within their summaries.



In the image above, this query will search for any issues containing the term "Team A" and add it to the structure.

Depending on your query, this could result in a large number of issues being added. You can limit that number by adjusting the **Limit** field. By default, the limit is set at 1,000 issues. If you don't want to limit the number of issues, simply leave this field blank.

Once you've entered your query and set your Limit, click **Apply**. The inserter should now appear as a new row in your structure, with the added issues placed below it.



When to Use the Text Query Inserter

The Text Query Inserter is particularly useful if your company uses specific naming conventions, or if you need to find similar issues across multiple projects.

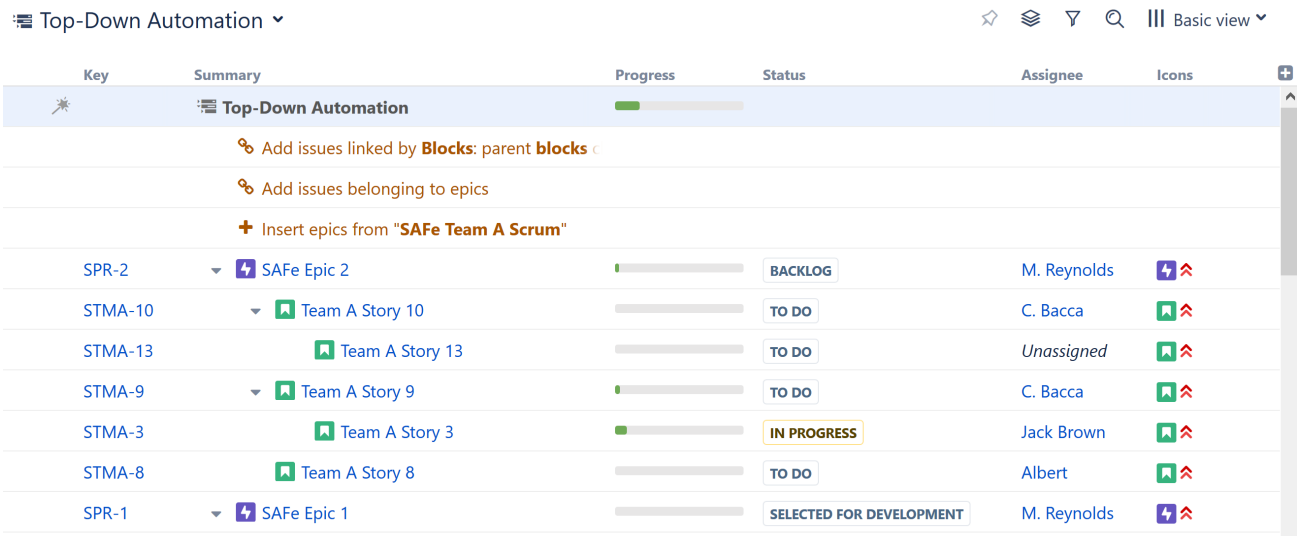
For example:

- You need to track issues related to a specific application, which may be used for several projects
- You need to track issues involving a specific partner or client
- Your team uses specific terms to designate an internal issue property

You can expand and customize your results using the Extend(see page 148), Filter(see page 158), Sort(see page 147) and/or Group(see page 169) generators.

3.4.6 Extend Generators

Extend generators allow you to add issues to a structure based on Issue Links, Epic Links and Sub-task relationships.



Extend generators are often used in conjunction with an Insert generator(see page 142). In the example above, we used the Agile Board Insert(see page 145) to add our epics, followed by the Stories Under Epics Extender(see page 154) to add the stories beneath each epic and, finally, the Linked Issues Extender(see page 149) to add linked issues beneath these stories.

3.4.6.1 Types of Extenders

The following Insert generators are available:

- Linked Issues Extender(see page 149) — Pulls in issues that are linked to issues already in the structure.
- Sub-tasks Extender(see page 152) — Pulls in sub-tasks to issues already in your structure.
- Stories Under Epics Extender(see page 154) — Pulls in issues belonging to epics already in your structure.
- Child Issues (Advanced Roadmaps) Extender(see page 155) — Pulls in child issues using the Parent Link field from the Advanced Roadmaps for Jira add-on (if installed).
- Pages Extenders(see page 156) — Pulls in child pages or linked pages from Confluence (when the Structure.Pages add-on is installed).

Additional extenders may be available, based on other add-ons you have installed.

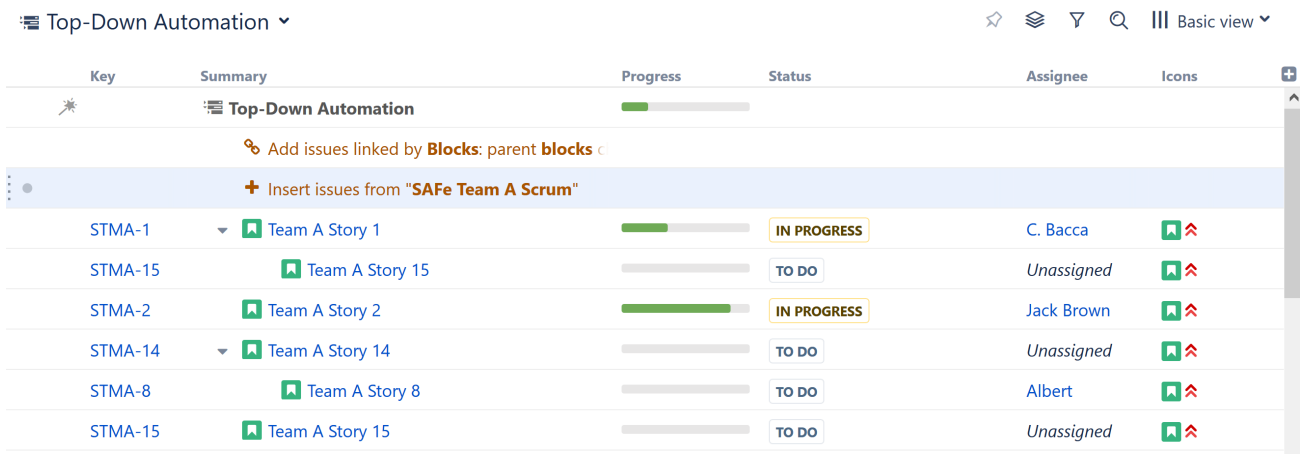
3.4.6.2 Always Up to Date

Generators run every time you open a structure, so the list of issues added by the Extend generator is always up to date.

Additionally, if issues change as you work with the structure, they will be added, removed, or moved accordingly, based on the rules of your Extend generator.

3.4.6.3 Linked Issues Extender

The Linked Issues Extender pulls in issues that are linked to issues already in the structure. Linked issues will be placed beneath the current issues in the structure's hierarchy.

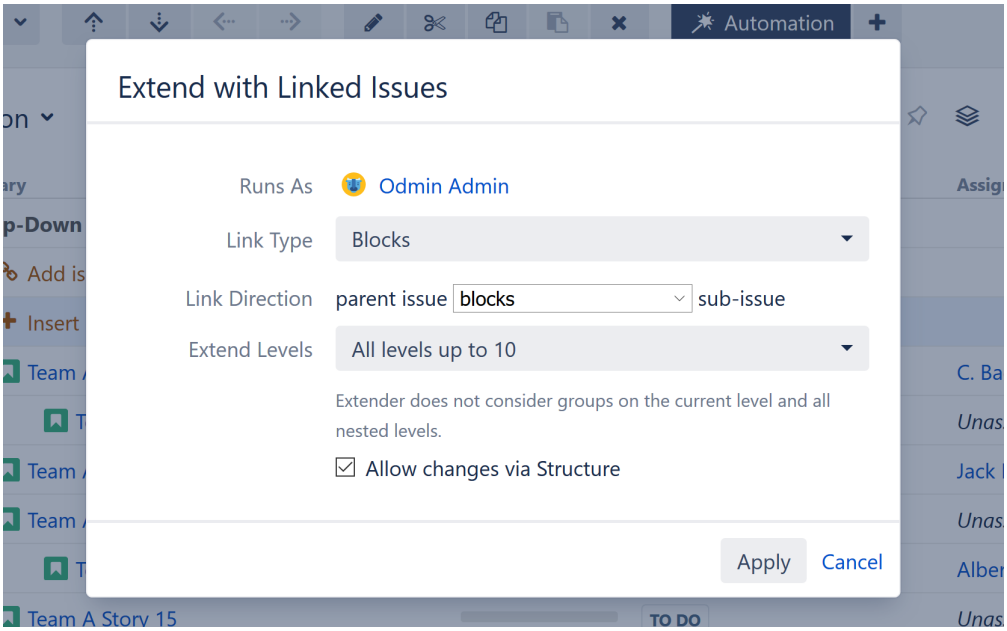


When using the Linked Issues Extender, some issues may appear in your structure more than once. In the example above, "Team A Story 15" appears twice, because it met the criteria for the original Inserter AND it was linked to "Team A Story 1."

You can use the Inserter/Extender Duplicates Filter(see page 162) to remove these duplicates from your structure.

Customize Your Extender

Each Linked Issues Extender can be customized to create exactly the hierarchy you need.



You can customize:

Link Type - Allows you to specify which links to add to your structure.

Link Direction - Defines which side of the link is the parent issue and which is the sub-issue.

Extend Levels - Allows you to select which levels in the hierarchy the extender should be applied to:

- *All levels up to 10* (default) - The extender will be applied to the first 10 levels of the hierarchy, starting from the level where the extender itself is located.
- *All levels* - The extender will be applied to the current level and all its descendants.
- *Current level only* - The extender will only be applied to the same level in the hierarchy as the extender itself.
- *Manual levels range* - You can specify which levels the extender is applied to.

See Generator Scope(see page 198) to learn more about customizing levels.

Allow changes via Structure - If this option is checked, links will be updated as you move issues in your structure:

- Moving a linked issue from beneath one issue to another will sever the original link and create a new link.
- Deleting a linked issue from the structure will sever its link.
- Copying an issue under another issue will create a new link.

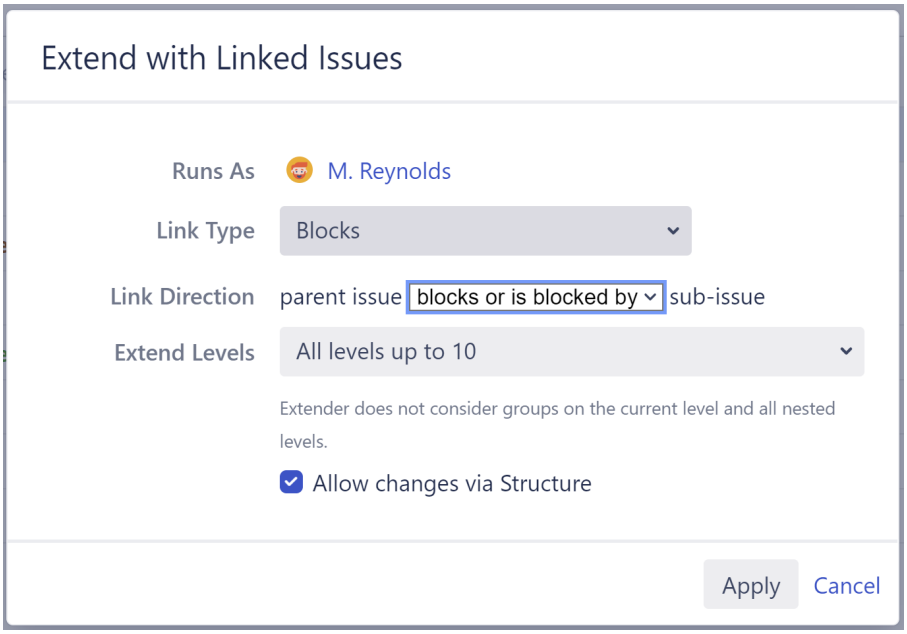
Runs As

When a generator runs, it runs as the structure owner. This is important because the generator will have access to the same projects, issues, etc. as the user listed here – if they don't have permission to view an issue, it won't be added to the structure.

Bi-directional Links

If you select the links to go in both directions (ex. "parent issue blocks or is blocked by sub-issue"), issues will be extended in the "shortest path" mode. This means linked issues will only be added the first time that link is represented in the structure.

This is done to prevent a strain on system resources and keep structures more manageable.



Here's what that looks like in a structure:

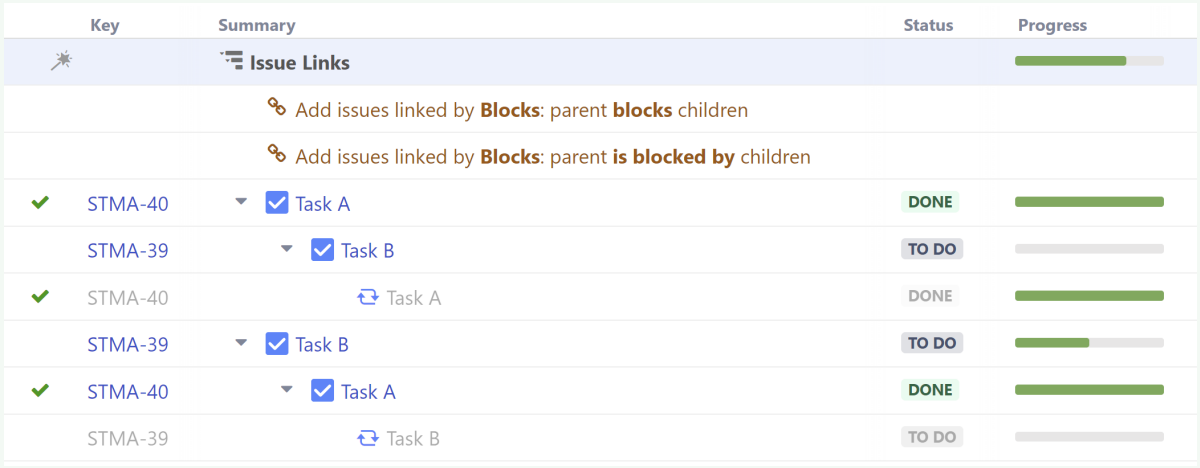
1. You add an Extend generator with the following settings:
   - Link Type: Blocks
   - Link Direction: parent issue blocks or is blocked by sub issue
   - Extend Levels: All levels up to 10
2. You have 2 issues, linked as follows: **Task A** blocks **Task B**.
3. **Task A** and **Task B** are added to the first level of your hierarchy.

The following hierarchy will is created:

	Key	Summary	Status	Progress
		Issue Links		
		Add issues linked by **Blocks**: parent **blocks or is blocked by** child		
✔	STMA-40	Task A	DONE	
	STMA-39	Task B	TO DO	
	STMA-39	Task B	TO DO	
✔	STMA-40	Task A	DONE	

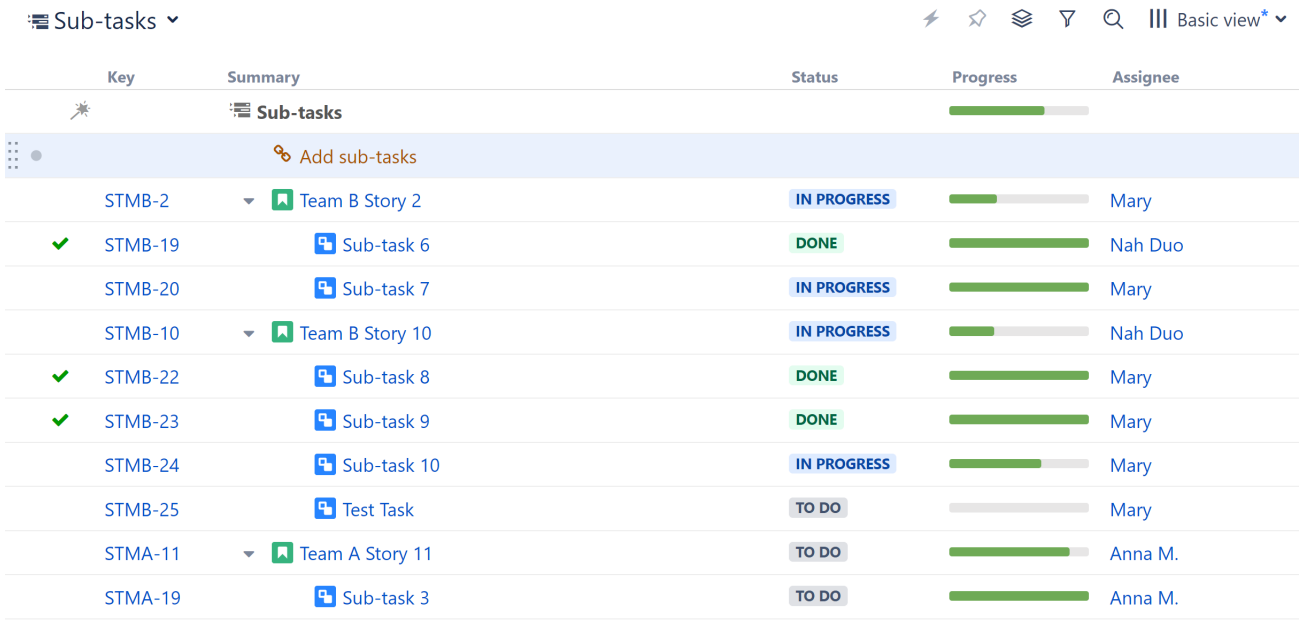
If the extender did not use "shortest path" mode, this hierarchy could continue in a loop, adding a third level of hierarchy (with **Task A** below **Task B** and **Task B** below **Task A)**, and then a fourth level, etc.... until the tenth level was reached.

If you prefer to visualize these loops, instead of adding a bi-direction Link extender, add 2 one-direction Link extenders. In this case, an additional level will be created in the hierarchy, indicating the loop:



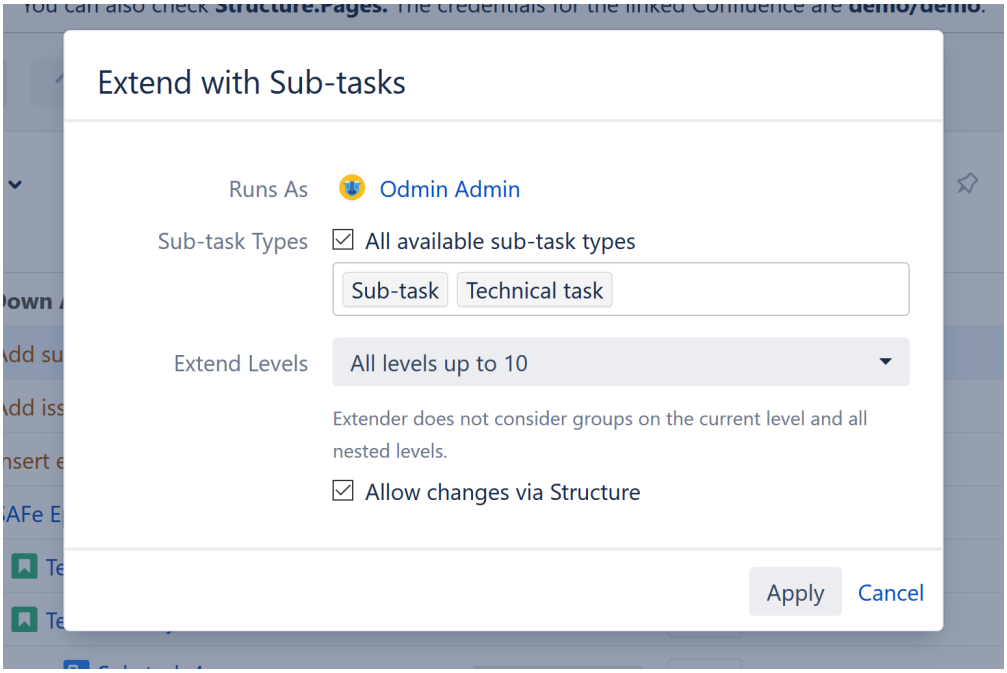
3.4.6.4 Sub-tasks Extender

The Sub-tasks Extender pulls in sub-tasks belonging to issues already in the structure. The sub-tasks will be placed beneath the issues in your hierarchy.



Customize Your Extender

Each Sub-tasks Extender can be customized to create exactly the hierarchy you need.



You can customize:

Sub-task Types - Allows you to specify which types of sub-tasks should be included in your structure. To include all sub-tasks, check the "All available sub-task types" box.

Extend Levels - Allows you to select which levels in the hierarchy the extender should be applied to:

- *All levels up to 10* (default) - The extender will be applied to the first 10 levels of the hierarchy, starting from the level where the extender itself is located.
- *All levels* - The extender will be applied to the current level and all its descendants.
- *Current level only* - The extender will only be applied to the same level in the hierarchy as the extender itself.
- *Manual levels range* - You can specify which levels the extender is applied to.

See Generator Scope(see page 198) to learn more about customizing levels.

Allow changes via Structure - If this option is checked, sub-tasks will be assigned to new parents as you move them in your structure.

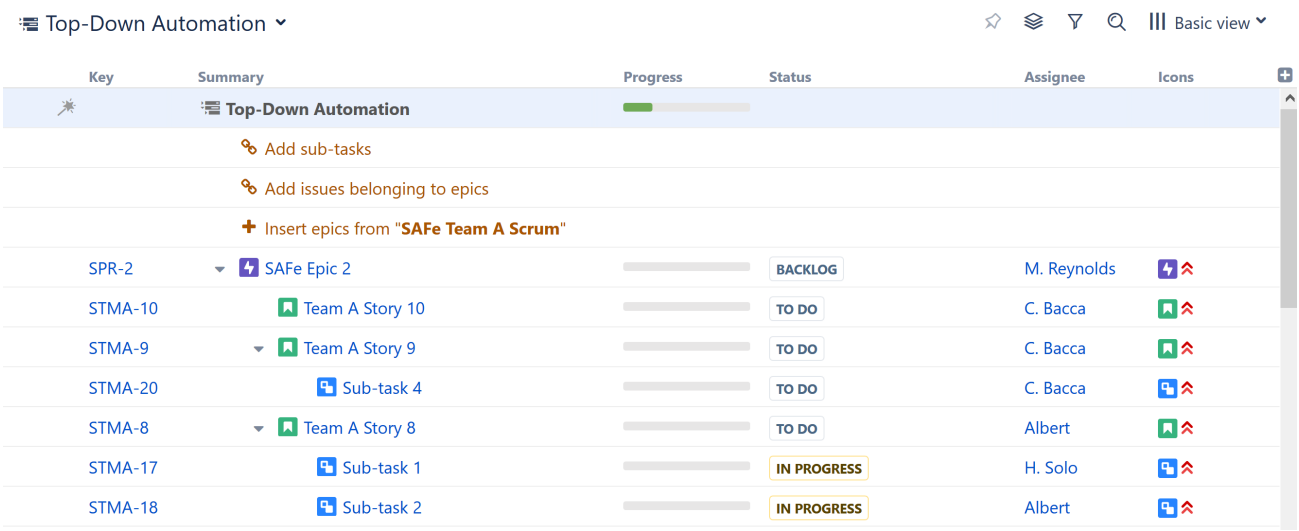
Runs As

When a generator runs, it runs as the structure owner. This is important because the generator will have access to the same projects, issues, etc. as the user listed here – if they don't have permission to view an issue, it won't be added to the structure.

Example

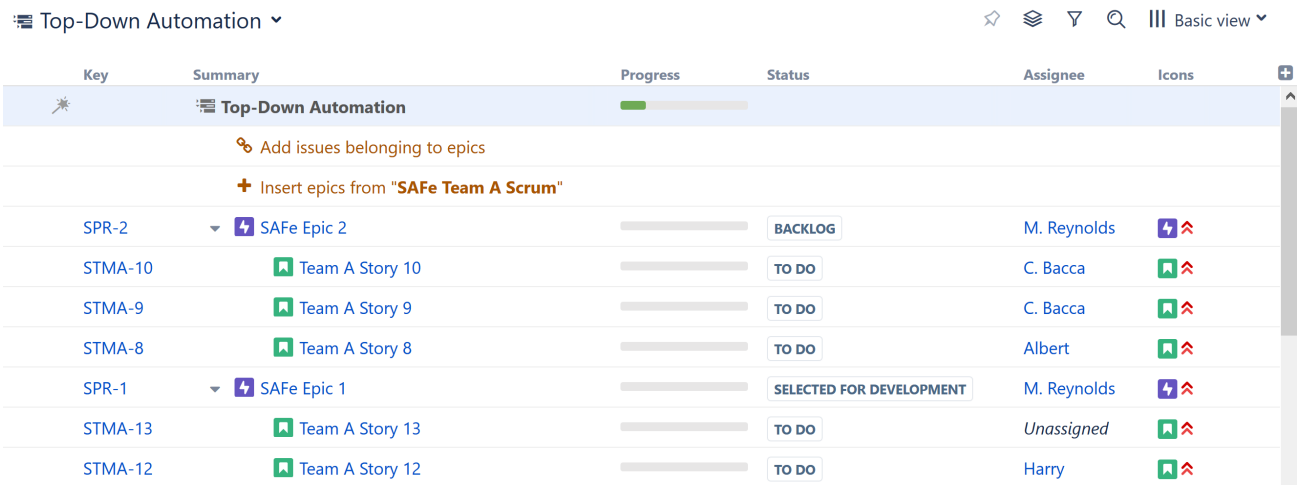
The Sub-tasks Extender can be used with other generators to build complex hierarchies. For example, to build an Epic > Story > Sub-task hierarchy:

1. Add all your epics with Agile Board Insert(see page 145)
2. Place issues beneath them, using the Stories Under Epics Extender(see page 154)
3. Place sub-tasks beneath issues, using the Sub-tasks Extender



3.4.6.5 Stories Under Epics Extender

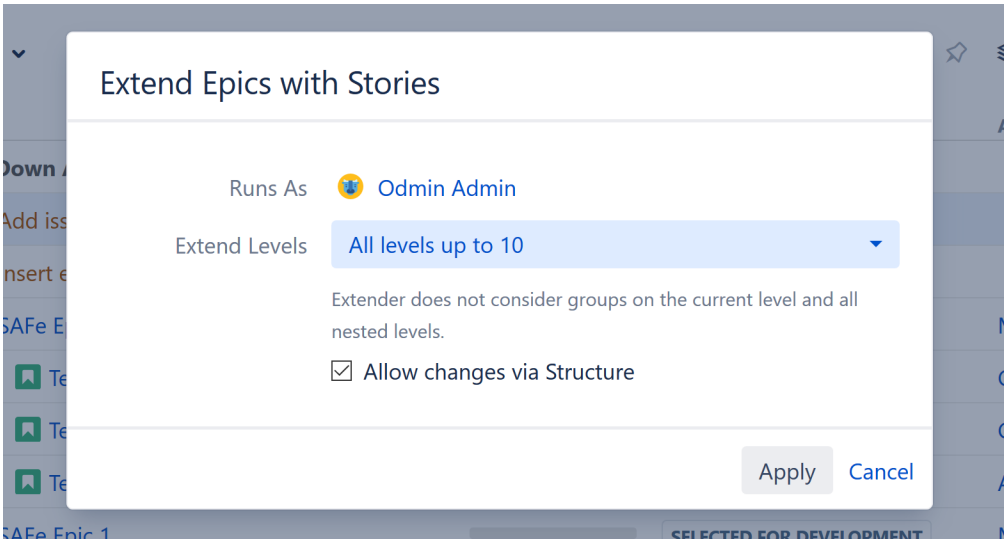
The Stories Under Epics Extender pulls in issues belonging to epics already in the structure. The issues will be placed beneath the epics in your hierarchy.



In the example above, we built our hierarchy by adding all our epics with an Agile Board Insert(see page 145), and then we placed our issues beneath them, using the Stories Under Epics extender.

Customize Your Extender

Each Stories Under Epics Extender can be customized to create exactly the hierarchy you need.



You can customize:

Extend Levels - Allows you to select which levels in the hierarchy the extender should be applied to:

- *All levels up to 10* (default) - The extender will be applied to the first 10 levels of the hierarchy, starting from the level where the extender itself is located.
- *All levels* - The extender will be applied to the current level and all its descendants.
- *Current level only* - The extender will only be applied to the same level in the hierarchy as the extender itself.
- *Manual levels range* - You can specify which levels the extender is applied to.

See Generator Scope(see page 198) to learn more about customizing levels.

Allow changes via Structure - If this option is checked, moving an issue from beneath one epic to another will update the issue's epic link in Jira. Deleting an issue from the structure will remove the epic link.

Runs As

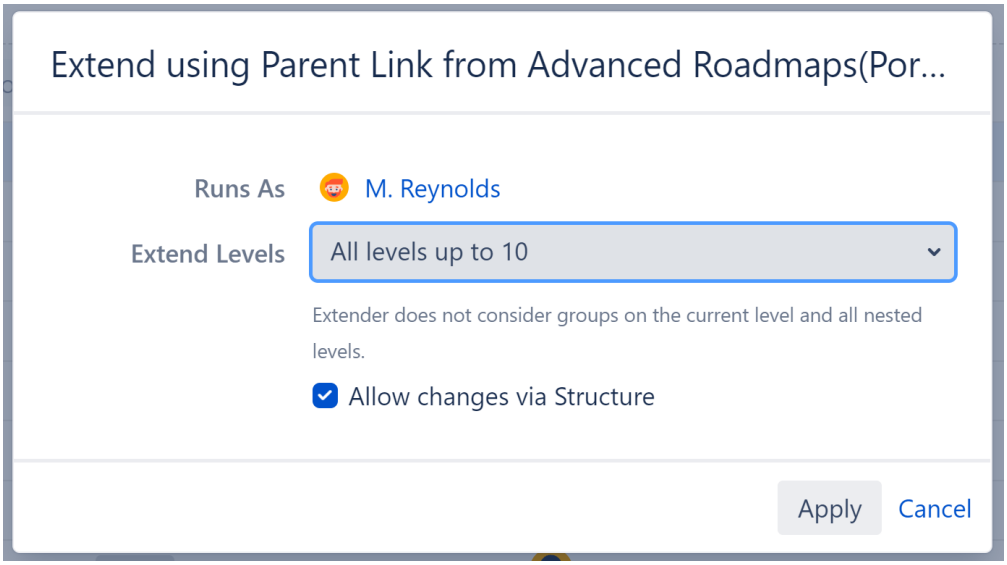
When a generator runs, it runs as the structure owner. This is important because the generator will have access to the same projects, issues, etc. as the user listed here – if they don't have permission to view an issue, it won't be added to the structure.

3.4.6.6 Child Issues (Advanced Roadmaps) Extender

If you have the Advanced Roadmaps for Jira (formerly Portfolio for Jira) add-on installed, the Child Issues (Advanced Roadmaps) Extender allows you to pull in child issues using Advanced Roadmaps' Parent Link field.

Customize Your Extender

Each Child Issues (Advanced Roadmaps) Extender can be customized to create exactly the hierarchy you need.



You can customize:

Extend Levels - Allows you to select which levels in the hierarchy the extender should be applied to:

- *All levels up to 10* (default) - The extender will be applied to the first 10 levels of the hierarchy, starting from the level where the extender itself is located.
- *All levels* - The extender will be applied to the current level and all its descendants.
- *Current level only* - The extender will only be applied to the same level in the hierarchy as the extender itself.
- *Manual levels range* - You can specify which levels the extender is applied to.

See Generator Scope(see page 198) to learn more about customizing levels.

Allow changes via Structure - If this option is checked, the Portfolio parent links will be updated as you move issues in your structure.

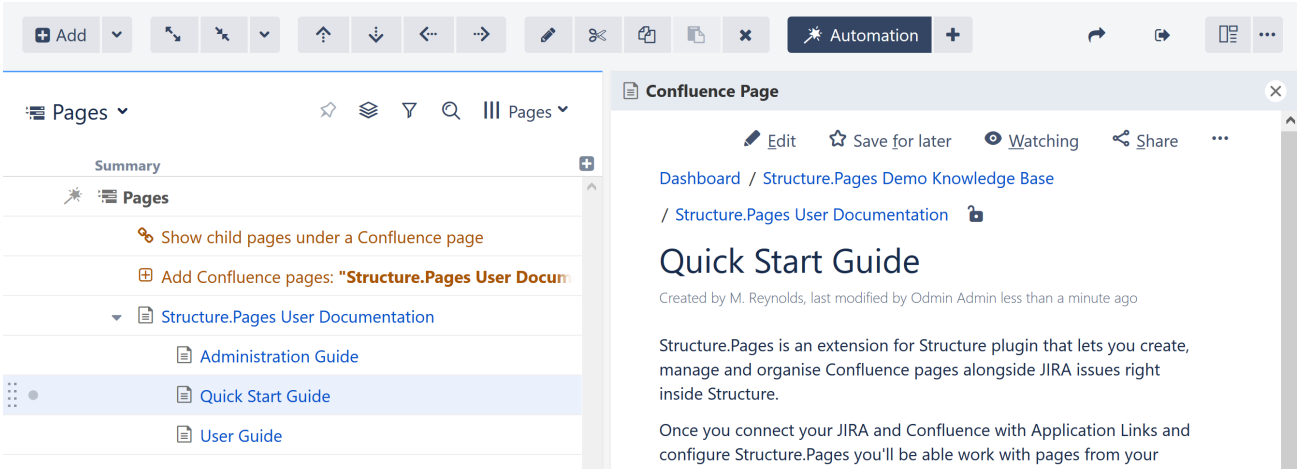
Runs As

When a generator runs, it runs as the structure owner. This is important because the generator will have access to the same projects, issues, etc. as the user listed here – if they don't have permission to view an issue, it won't be added to the structure.

3.4.6.7 Pages Extenders

If you have the Structure.Pages[31] add-on installed, you will see some additional extenders available specifically for Confluence pages.

31 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence



Child Pages Extender

The Child Pages Extender adds child pages beneath pages already in your structure.

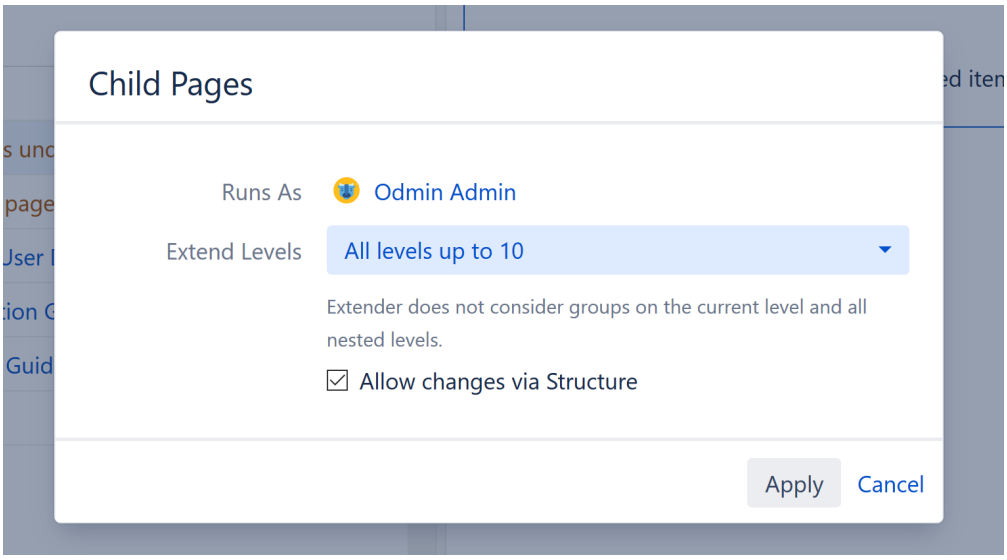
You can add pages manually, or using the Insert(see page 142) generator or Linked Pages Extender.

Linked Pages Extender

The Linked Pages Extender will pull in Confluence pages linked to issues in your structure.

Customize Your Extender

Each Pages extender can be customized to create exactly the hierarchy you need.



You can customize:

Extend Levels - Allows you to select which levels in the hierarchy the extender should be applied to:

- *All levels up to 10* (default) - The extender will be applied to the first 10 levels of the hierarchy, starting from the level where the extender itself is located.
- *All levels* - The extender will be applied to the current level and all its descendants.
- *Current level only* - The extender will only be applied to the same level in the hierarchy as the extender itself.
- *Manual levels range* - You can specify which levels the extender is applied to.

See Generator Scope(see page 198) to learn more about customizing levels.

Allow changes via Structure - If this option is checked, moving pages within the structure will update their location in Confluence or their links within Jira.

Runs As

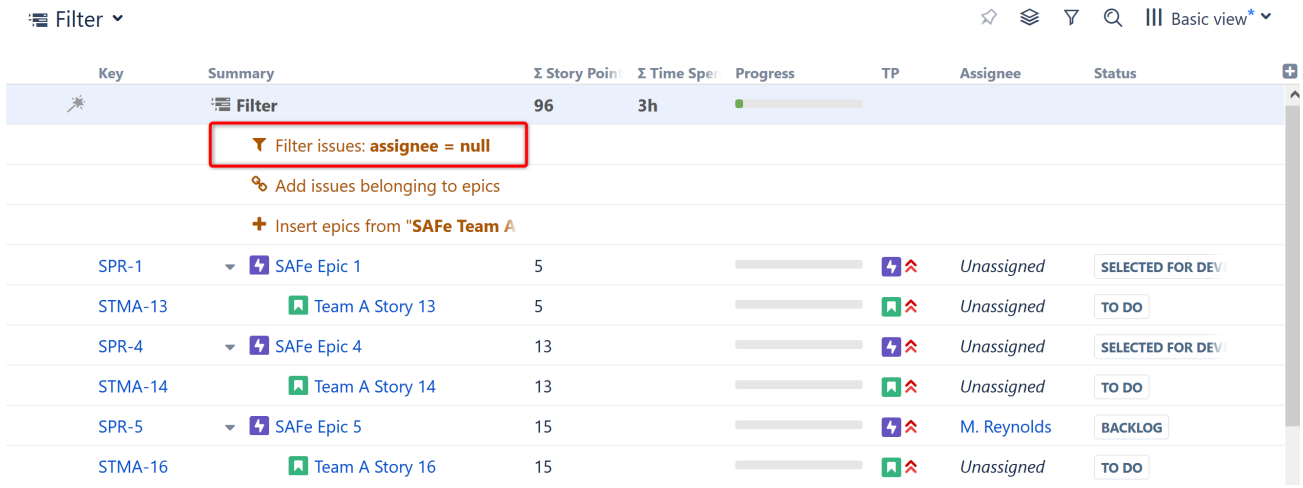
When a generator runs, it runs as the structure owner. This is important because the generator will have access to the same projects, issues, etc. as the user listed here – if they don't have permission to view an issue, it won't be added to the structure.

3.4.7 Filter Generators

Filter generators allow you to limit the scope of your structure by removing any issues that do not pass certain criteria. Structure offers the following types of filters:

- Filter by Field Generator(see page 167)
- Hide Closed Sprints
- Remove Inserter/Extender Duplicates(see page 162)
- JQL Query
- S-JQL(see page 428) Query
- Sprints - filter by active, future, or past sprints
- Text Query
- Attribute (see page 160)- filter issues based on the values of a specified attribute

Once you add this generator, you will only see those items that pass the query and their ancestors.



The ancestors are necessary to show the issues' placement within the hierarchy.

⚠ Placement matters. Generators only affect issues beneath them, so if you want to filter the entire structure, place the generator at the very top (by selecting the structure's name in the top row). If you place it anywhere else, it will only filter the items beneath it.

3.4.7.1 When to Use the Filter Generator

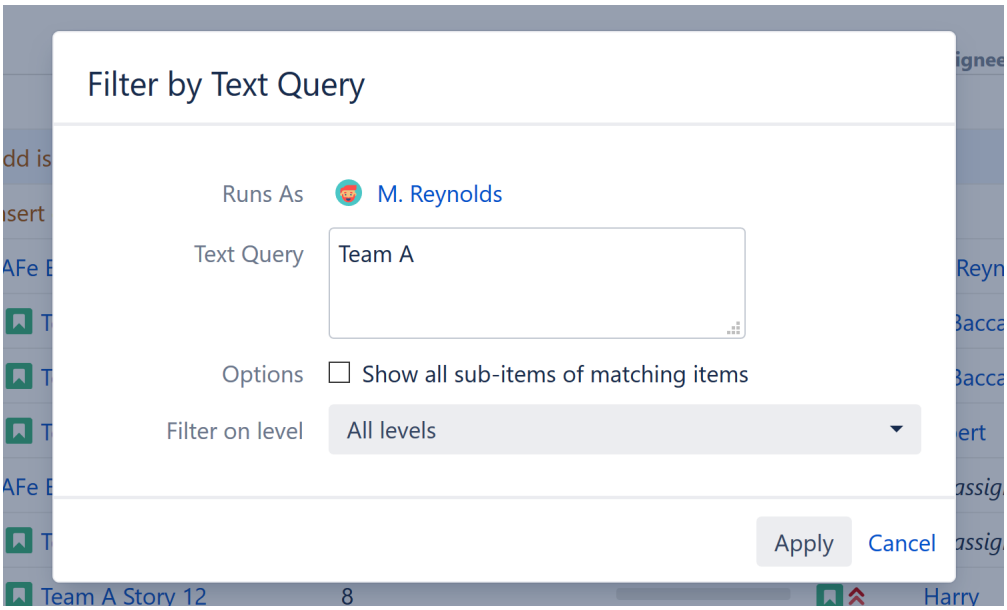
The Filter generator can be used with a manually-built structure, or combined with an Insert generator(see page 142) to limit the issues that are added.

It is particularly useful when used in conjunction with other generators, in order to limit the number of issues based on a specific Jira or Structure attribute – in the example above, we've filtered out everything except unassigned issues.

✓ Use the Filter generator when you always want issues filtered within the structure. To temporarily filter issues, use the Filter(see page 207) button in the toolbar or Quick Transformations(see page 220).

3.4.7.2 Configuring a Filter

When setting filters, you can select certain options to customize which issues wind up in your structure.



While each filter has its own options, they all include the following:

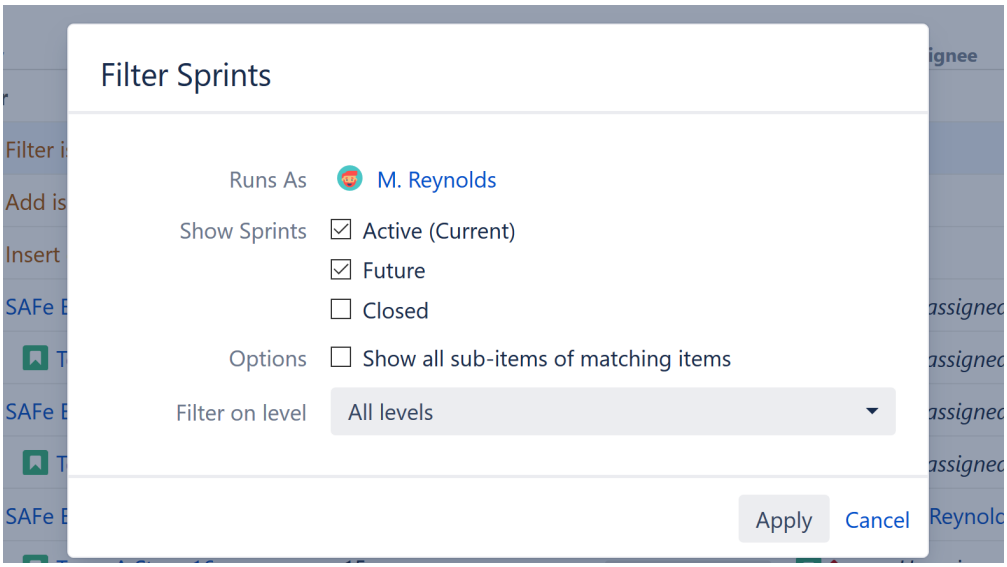
- **Show all sub-items of matching items** - If this option is selected, all issues that match your filter criteria will be included in the structure, along with any sub-items of those issues.
- **Filter on level** - You can apply a filter to specific levels within your hierarchy. For example, you may want to include all top-level items, but then filter the stories beneath them. See Generator Scope(see page 198) to learn more about customizing levels.

Runs As

When a generator runs, it runs as the structure owner. This is important because the generator will have access to the same projects, issues, etc. as the user listed here – if they don't have permission to view an issue, it won't be added to the structure.

Filter by Sprints

With the Sprints filter, you can select whether to include active, future, and/or closed sprints.



The Hide Closed Sprints filter does not offer any options when first selected, but it is in fact a Sprints filter with only the Active and Future sprints selected.

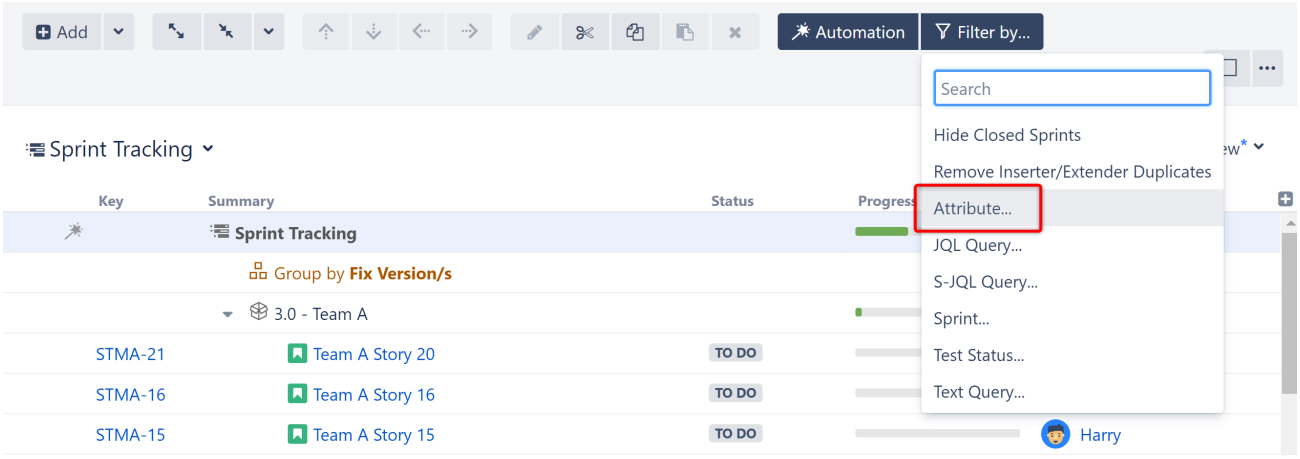
Filtering by Sprint (and the predefined Hide Closed Sprints) is limited to inserted structure(s)(see page 146), not to the whole structure. Also, as these filters apply to Sprint folders rather than issues themselves, Show all sub-items of matching items is made redundant.

Keep non-issues

This option is available under the JQL filter. When checked, non-issue items, such as folders, will remain in your structure regardless of whether or not they match your filter criteria.

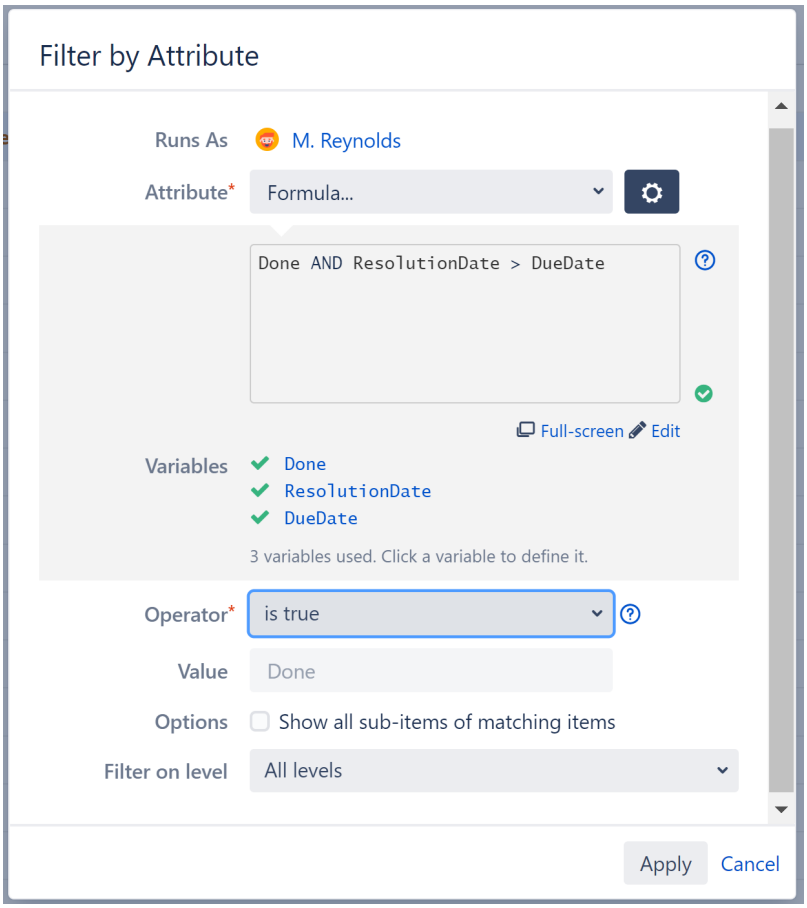
3.4.7.3 Attribute Filter

The Attribute filter allows you to limit the issues in a structure based on the values in a specific attribute. To apply an Attribute filter to a structure, open the Automation menu, select filter, and choose the Attribute filter.



On the settings page, select the values for the filter:

- Attribute - The attribute can be a Jira field, Structure field or attribute, Structure column, or even a formula.
- Operator - The operator is used to compare values. Be sure to select an operators appropriate to the attribute value (string, numerical, date, duration, etc.).
- Value - The value used for comparison.



Once applied, Structure will compare each issue's attribute value against the operator/value combination:

- If the issue matches, it will remain in the structure.
- If the issue does not match, and has no sub-issues that match, it will be removed.

- If the issue does not match, but has sub-issues that do, the non-matching issue will remain but be grayed out.

The Attribute filter can be used with Generators(see page 140) (shown above) and Transformations(see page 215).

Your filter settings may look a little different from the one above. Because we used a formula(see page 228) as our attribute, the section in gray was added so we could enter our formula and assign variables.

Examples

The following are just a few of the many possible uses for the Attribute filter.

Try filtering a structure by:

- **Time in Status** - Use this attribute with the "duration longer" operator to identify issues that are taking too long in a particular status.
- **Formula** - Filter tasks based on the results of a formula (for example, filter based on your internal KPIs, or show only overdue items).
- **Text filter** - Filter a formula or notes column for specific words or strings.
- **Regex** - Filter by regular expressions using the "contains" operator.
- **Multi-value fields** - Use the "compare" operator to check all the values in the field against a single filter value, and keep any issues that contain a match.
- **Empty values** - Use the "is empty" operator to identify zero values in numeric fields, spaces, empty lines, zero duration (estimates, work logged), and not-set values (like Unassigned).
- **Gantt Attributes** - If you have Structure.Gantt[32] installed, you can also filter by Gantt attributes (such as the Gantt Start or Finish date, or issues in the Critical Path). *Note: Gantt attributes are only available when using the Filter by Attributes* Transformation(see page 215)*; Automation does not support Gantt attributes.*

Requirements and Suggestions

Please note the following when configuring an Attribute filter:

- **Duration attributes** - The value should be defined in Jira format (1w1d1h).
- **Progress attributes** - Progress values should be expressed as a fractional or decimal number from 0 to 1. *Note: a period should be used to express decimal values (ex. 0.5).*
- **Username fields** - To filter by a user-type attribute (Assignee, user-type custom field, formula that results in a username), the value entered must be the username, not the user's full name.
- Group generators(see page 169) **with an Attribute filter** - If a structure contains both types of generators, the filter will run before the grouper - this could result in unexpected changes to the structure, and some groups may disappear completely if the accompanying issues are filtered out. See Order of Operation for Generators(see page 204) for more information.

3.4.7.4 Inserter/Extender Duplicates Filter

When you build a structure using both Insert(see page 142) and Extend(see page 148) generators, there is a chance you will wind up with duplicate issues – the Inserter/Extender Duplicate filter allows you to quickly remove those duplicate issues from your structure.

Here's how this can happen:

32 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale?hosting=server&tab=overview

1. You use an Insert generator to add every Story from a current project. Those stories are added to the the top level of your structure.
2. You then use the Linked Issues Extender (see page 149)to add issues linked to those stories - these issues will be placed below your existing stories in the structure.
3. If any of the original stories are linked to each other, they will appear more than once in your structure – at the top level (because they were added by the inserter) and as children of other issues (because they were added again by the extender).

The Inserter/Extender Duplicate Filter will remove such issues from the top level and only keep the children. Please see examples below for a more detailed explanation.

Examples

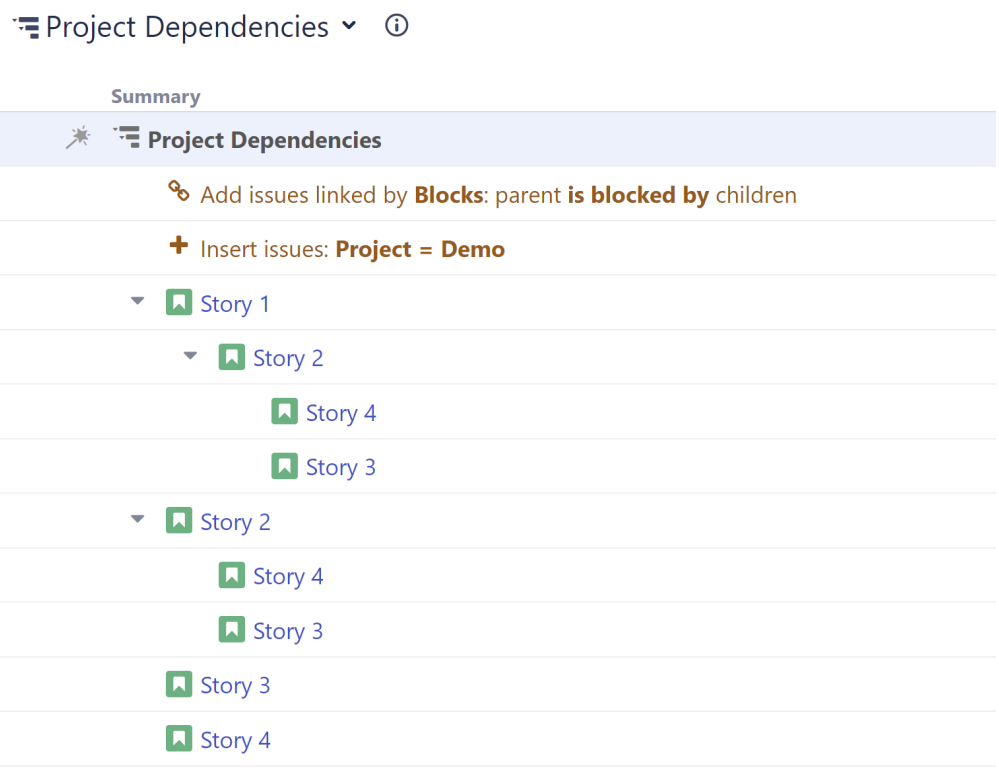
Basic Links

Imagine we have a project with issues Story 1, Story 2, Story 3, and Story 4, and some of the issues are blocking other issues:

- Story 1 is blocked by Story 2
- Story 2 is blocked by Story 3 and Story 4

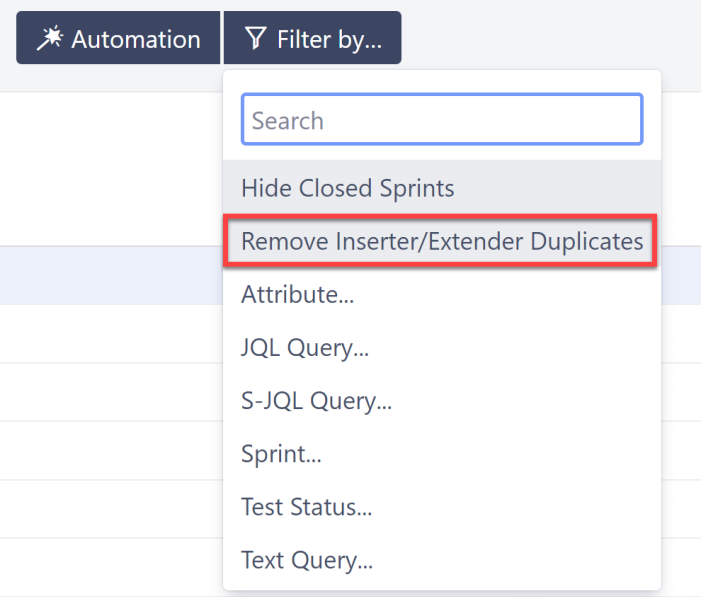
In our structure, we want to see all issues from our project arranged based on the existing "Blocks" links.

After you add all four issues by a JQL Inserter and add a Links Extender, you will get the following hierarchy:

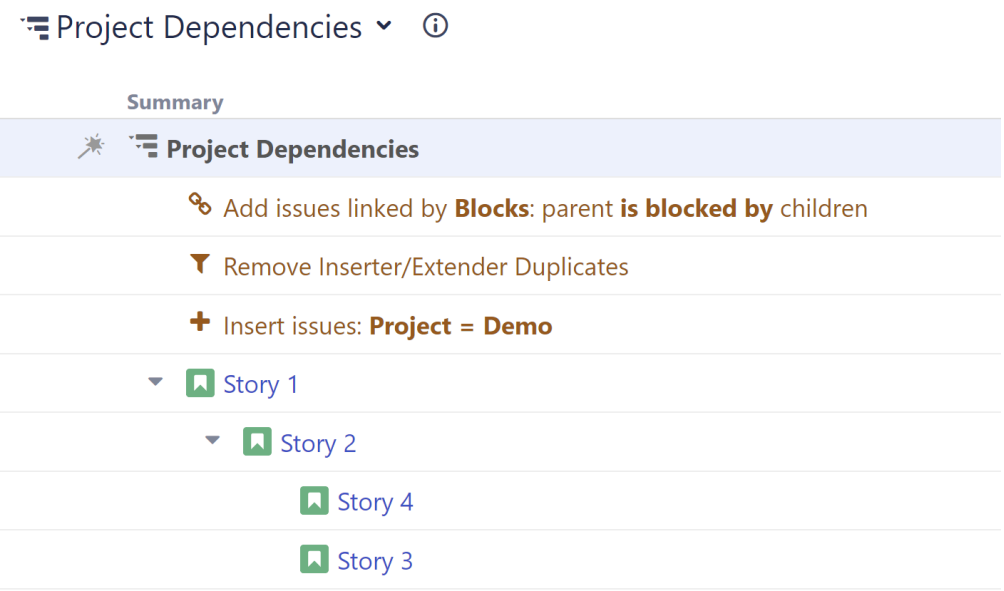


You can see that some issues have been duplicated, because the Extender adds the children under parents, even if they are already in the structure.

Now let's add the Duplicates Filter:



As a result, we get a structure with the hierarchy and no duplicates:

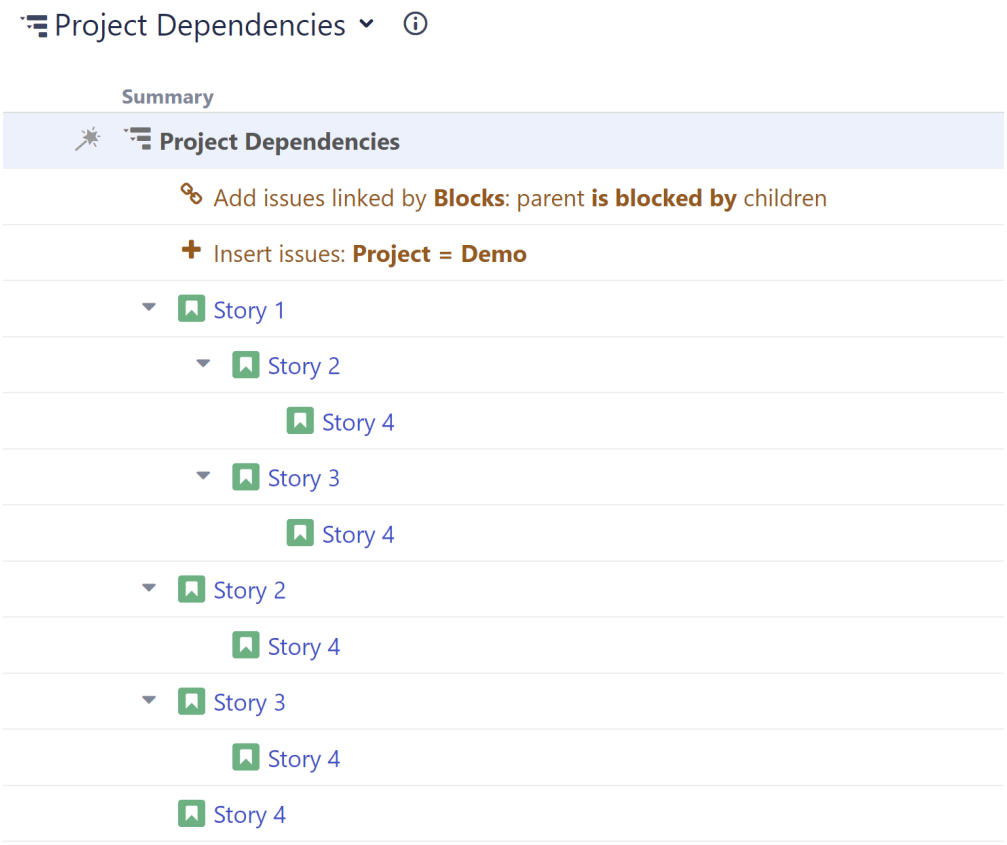


Multiple Parents

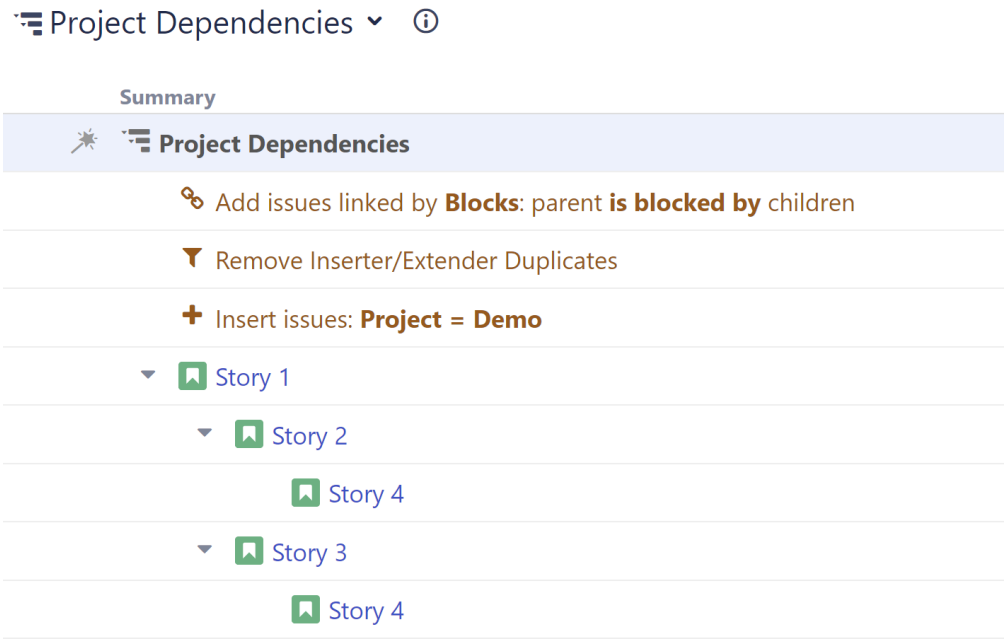
We have the same situation as in the example above, but we have one story that blocks two other stories, so it should be shown under both of them:

- Story 1 is blocked by Story 2 and Story 3
- Story 2 and Story 3 are blocked by Story 4

Without Duplicate Filter, it looks like this:



With the filter applied, any identical instances are removed:



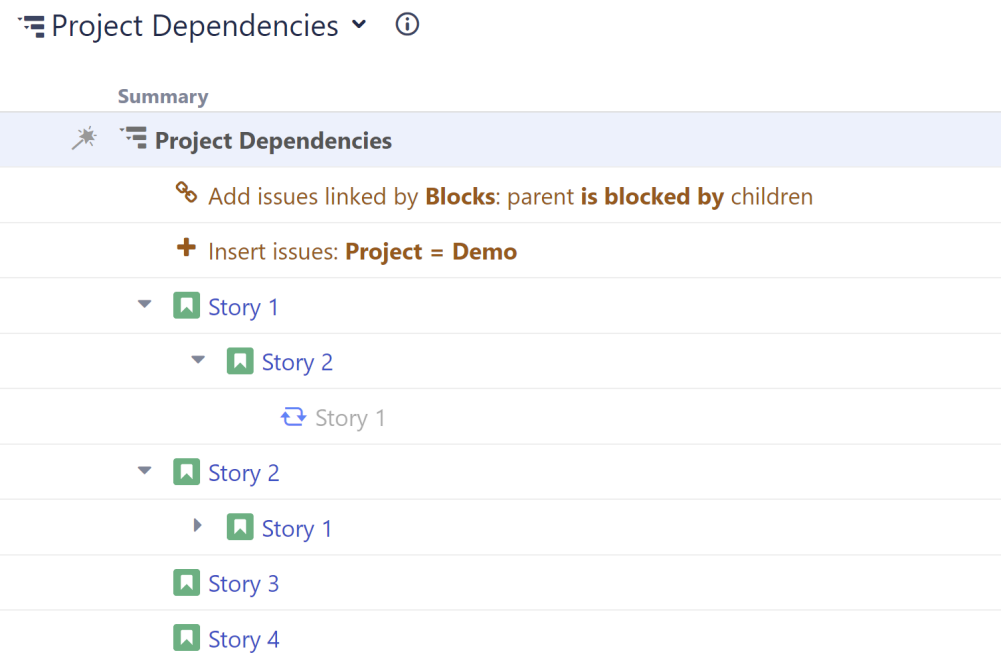
In this example, Story 4 still appears twice – this is because these two instances are not identical. One is blocking Story 2, while the other is blocking Story 3. Both need to be expressed in the hierarchy.

Link Cycles

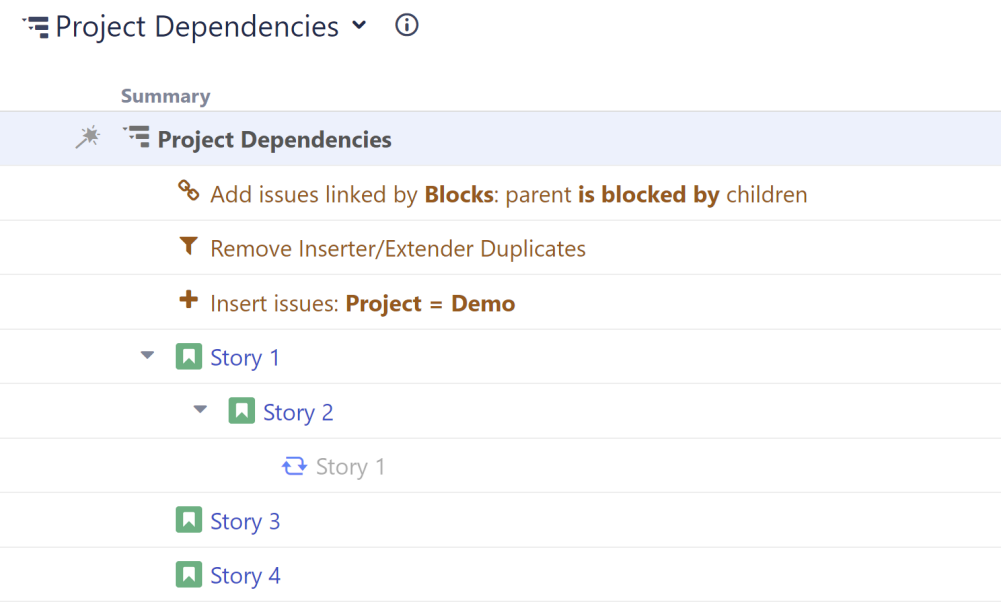
If there are link cycles between the issues (both issues link to each other), the Duplicates Filter will remove one of the branches and keep the other, to make sure all the issues added by the Insert and Extend generators are in the structure.

In this example, Story 1 blocks Story 2 and Story 2 blocks Story 1.

Without the filter, we get the following structure:

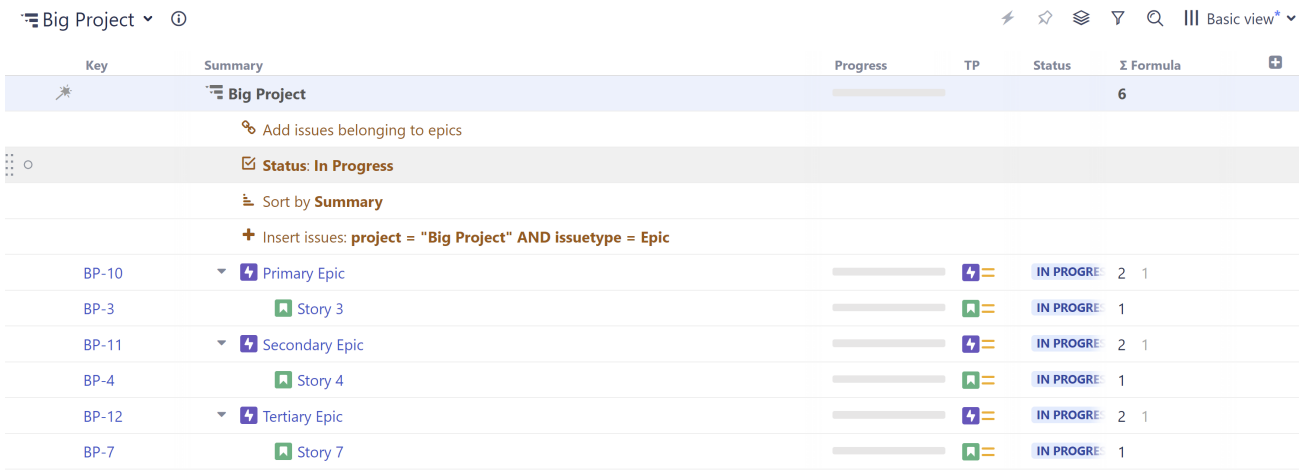


With the filter, one of the branches with the cycle gets removed:



3.4.7.5 Filter by Field Generator

The Filter by Field generator is the easiest way to filter a structure based on values in a Jira field. This allows you to focus on specific issues, such as issues from the next fix version, issues assigned to certain users, issues in a particular status, and more.



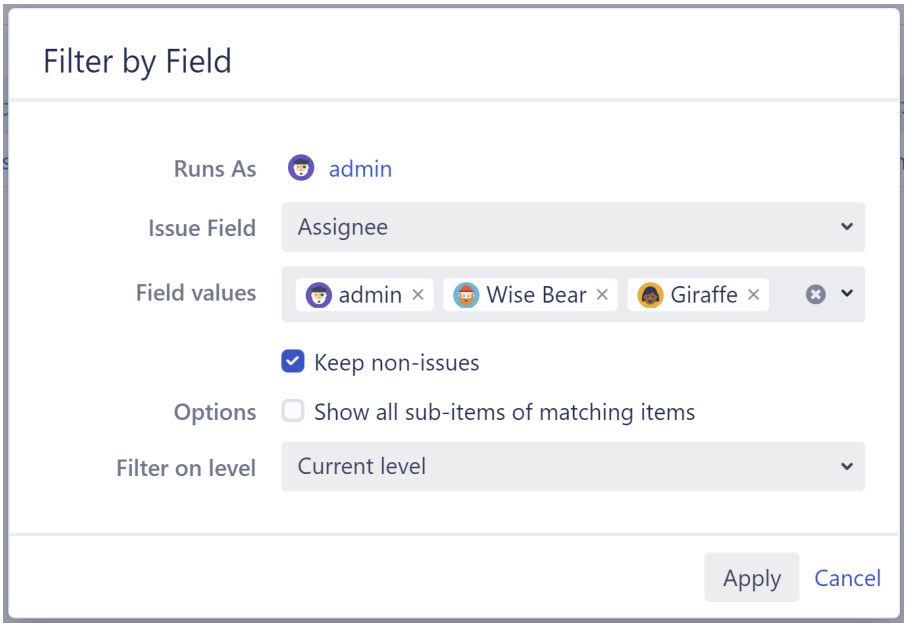
Once the generator is added, you will only see those items with the field values you've specified and their ancestors.

> The ancestors are necessary to show the issues' placement within the hierarchy.

> Placement matters. Generators only affect issues beneath them, so if you want to filter the entire structure, place the generator at the very top (by selecting the structure's name in the top row). If you place it anywhere else, it will only filter the items beneath it.

Configuring a Filter by Field Generator

When setting filters, you can select certain options to customize which issues wind up in your structure.



While each filter has its own options, they all include the following:

- **Runs As -** When a generator runs, it runs as the structure owner. This is important because the generator will have access to the same projects, issues, etc. as the user listed here – if they don't have permission to view an issue, it won't be added to the structure.
- **Keep non-issues** - When checked, non-issue items, such as folders, will remain in your structure regardless of whether or not they match your filter criteria.
- **Issue Field** - Select the field that will be used to filter issues.
- **Field Values** - Any issues with the field values selected here will remain in the structure.
- **Show all sub-items of matching items** - If this option is selected, all issues that match your filter criteria will be included in the structure, along with any sub-items of those issues.
- **Filter on level** - You can apply a filter to specific levels within your hierarchy. For example, you may want to include all top-level items, but then filter the stories beneath them. See Generator Scope(see page 198) to learn more about customizing levels.

Filter by Field Transformation vs Generator

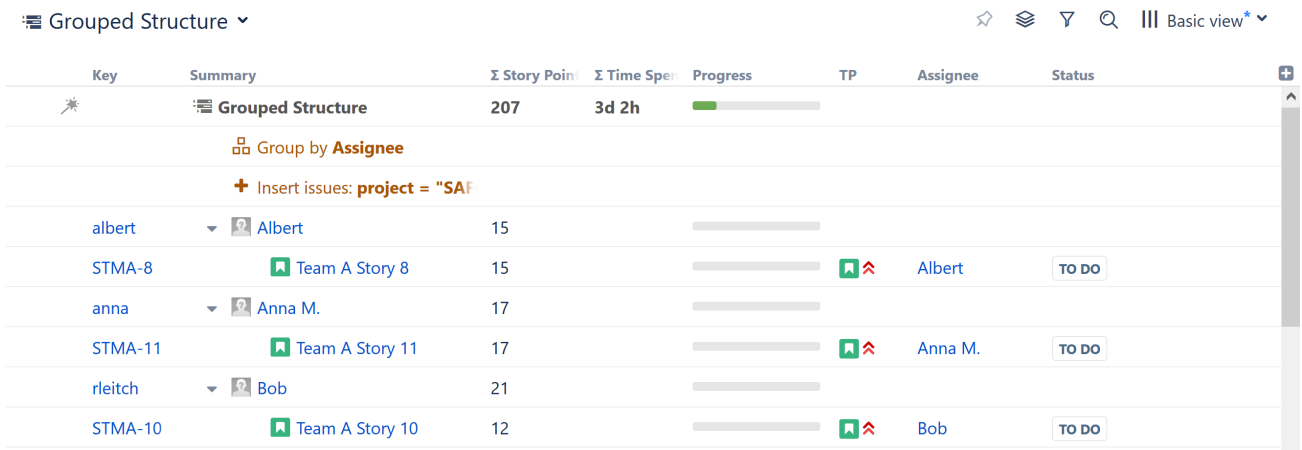
Filter by Field is available as a Transformation and a Generator.

- The Filter by Field transformation(see page 213) allows you to temporarilly filter items in a structure to focus on very specific issues. The filter is applied locally, so anyone else viewing the structure is unaffected. This is useful when you need to focus in on specific issues without changing the underlying structure for yourself or anyone else.
- The Filter by Field generator is a more permanent solution, which should only be used when you always want to see just the filtered results in your structure. This filter will be applied every time the structure is opened, and when the structure is shared.

⚠ If you're using a Filter by Field generator and an Insert generator(see page 142), you should consider using a more specific Insert generator instead - for example, edit your JQL Insert generator to only include issues with the specific field results. This can improve overall performance, particularly for larger structures, because issues do not have to be added and then removed.

3.4.8 Group Generators

Group generators allow you to group issues by most standard Jira fields, custom fields, and other issue attributes.



Depending on the field you are using, the Group generator may simply organize the issues currently in your structure, or it may add new issues to your structure:

- Grouping by issue types or links will add a new level of parent issues. For example, when you add a Group by Epics generator, issues will be grouped by their corresponding epics.
- Grouping by other fields will create new folders and place your current items into them. For example, Group by Assignee creates new folders for each assignee (see above).

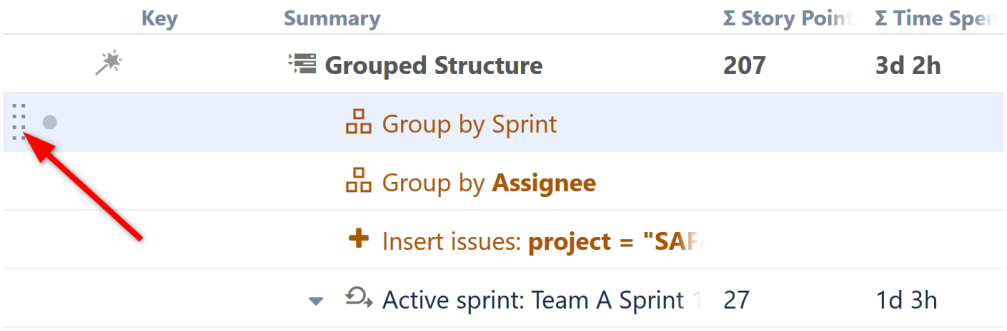
> ⚠ Placement matters. Generators only affect issues beneath them, so if you want to group the entire structure, place the generator at the very top (by selecting the structure's name in the top row). If you place it anywhere else, it will only group the items beneath it.

3.4.8.1 Grouping Order

Generators are applied to the current level, in the order they appear in the structure.

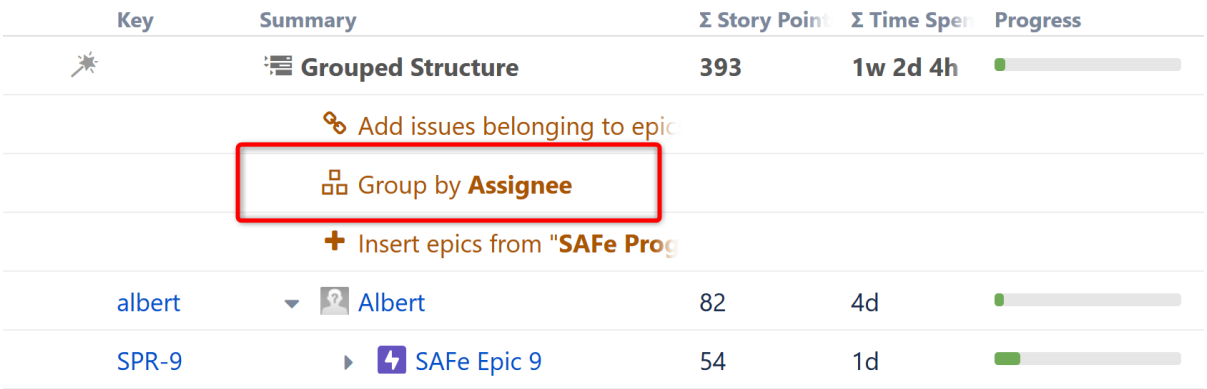
In the example below, we used an Insert generator to add issues, and then added Group by Sprint and Group by Assignee generators. Two new levels were created, grouping our issues first by Sprints and then by Assignees:

To rearrange the hierarchy so that Assignees are at the top level, followed by Sprints, simply reorder the generators. In this case, drag-and-drop **Group by Sprint** under **Group by Assignee**.

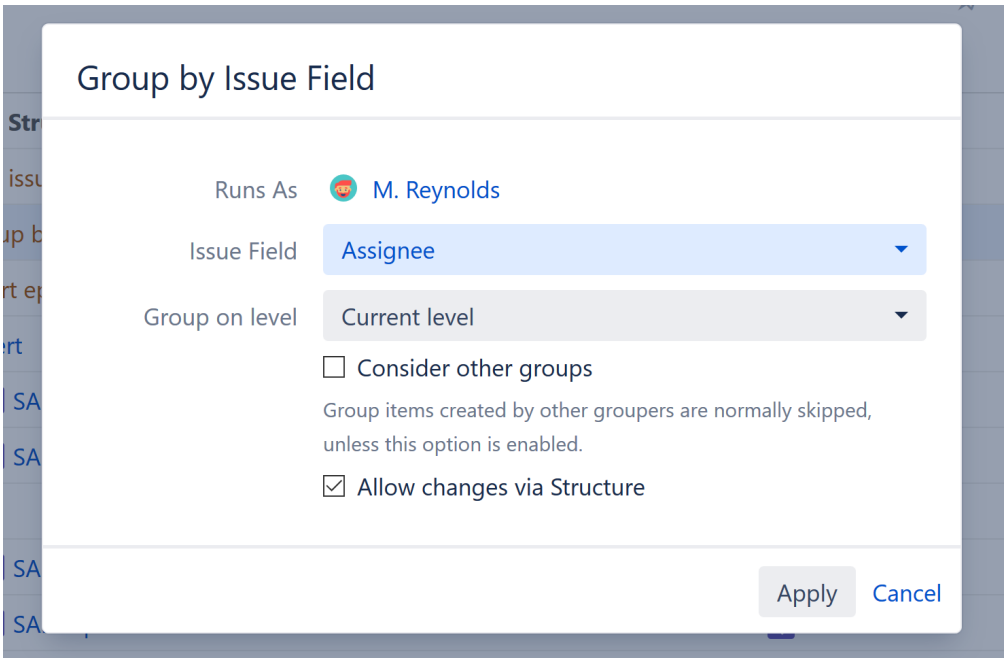


3.4.8.2 Customize Your Grouping

Most of the options under the "Group By..." Automation are applied the moment they are selected, and you will not be asked to set parameters for the group. However, you can still customize a Group generator by locating it within your structure and double-clicking its summary.



From the options dialogue, you can select a new issue field to group your structure by, change the level within your hierarchy where the sort is applied and more.



- **Group on level**- You can apply the grouping to the current level, next level, or manually enter a level within the hierarchy. See Generator Scope(see page 198) to learn more.
- **Consider other groups**- By default, a new Group generator will ignore data created by other Group generators.
- **Allow changes via Structure** - If this option is checked, you can update an issue's field simply by dragging it to a new group. For examples, if you have grouped by assignee, moving an issue to another assignee group will reassign the issue.

⚠ Updating a field may not always be possible. You cannot change attributes(see page 175) or the results of a formula by moving issues within a structure. You also cannot transition an issue to a new status if there is an additional screen assigned to that transition.

If a move results in an invalid change, you will receive an error message, and the moved issue will return to its original location.

Consider other Groups

By default, Group generators ignore data added to your structure by other Group generators, because in most cases the added data doesn't group well.

Let's look at the scenario above, where we grouped our structure by both Assignee and Sprint. Here's a breakdown of how that works:

1. Structure starts with your top level issues (or adds them with an Insert generator(see page 142)).
2. Next, it applies the first Group generator in our list, the Group by Sprint. This creates a new level in your hierarchy.

3. Then Structure applies the Group by Assignee generator. At this point, the true "current" level is a list of sprints, which are not issues and don't have Assignees to group them by. So Structure ignores these items, and creates the new group based on the original issues.

There may be times when you want to include the results of a Group generator. For example, the Group by Issue Link generator adds a new level of issues to your structure. If you want to group the resulting issues by another attribute, select the **Consider other groups** option.

Runs As

When a generator runs, it runs as the structure owner. This is important because the generator will have access to the same projects, issues, etc. as the user listed here – if they don't have permission to view an issue, it won't be added to the structure.

3.4.8.3 Grouping Attributes

You can group a structure by any of the following attributes:

- **Standard fields**: such as Affects Version, Assignee, Component, Epic, Epic Status, Epic/Theme, Fix Version, Flagged, Issue Type, Labels, Priority, Project, Reporter, Resolution, Status, Sprint
- **Jira custom fields**: fields that give you a list of values to choose from, including radio button, list single choice, checkboxes, user picker, labels and select list
- **Attributes:** read-only values, formulas, and non-Jira fields - see Group by Attribute(see page 175)
- **Advanced Roadmaps parent link:** as defined in Advanced Roadmaps for Jira(formerly Portfolio for Jira)
- **Tempo Account:** as defined in Tempo for Jira
- **Issue links**: group issues by their linked issues. With this generator, you can select link type and direction. For example, you can group issues under their respective blockers (issues that block them).
- **Customer Request Type**: as defined in Jira Service Management (formerly Jira Service Desk)

3.4.8.4 Grouping by a Multiple-Selection Field

Issues can be grouped by attributes that allow multiple selections, such as Labels. This could result in issues appearing more than once in your structure.

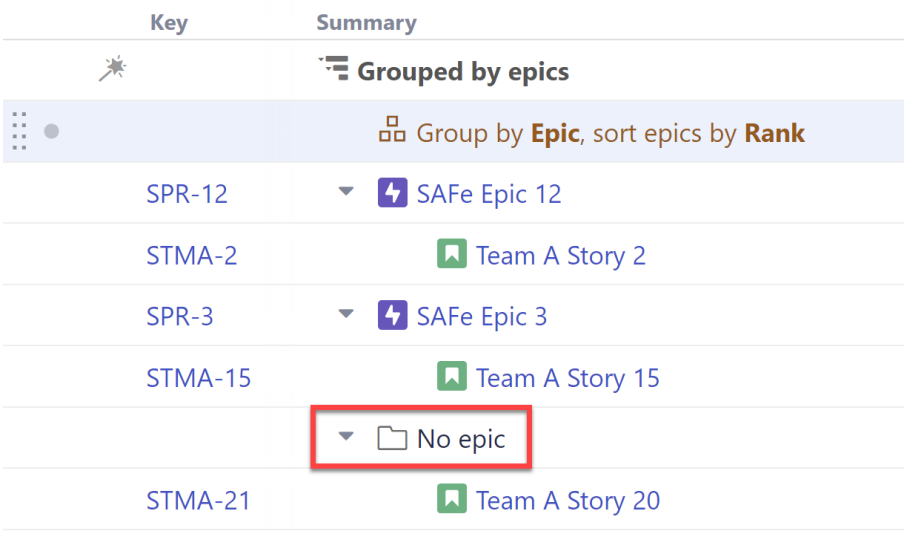
Additionally, if the **Allow changes via Structure** option is enabled, the following will happen when you move an issue between these groups:

- **Moving from one group to another** - This will remove the original value and add the new value.
- **Copying from one group to another** - This will add the new value, while keeping the original value.
- **Deleting the issue from a group** - This will remove the issue from the structure (if allowed), but will not remove the field value.

If any of the issues being grouped do not have a value in the group-by field, a "No x" folder will be created, where "x" is the name of the field. Moving an item to this folder will remove all values from the field.

3.4.8.5 "No Value" Groups

When items in the structure do not have values for the group, they will be placed in a special folder at the buttom of the grouping level, such as "No epics" (when grouping by epics) or "Unassigned" (when grouping by assignee).



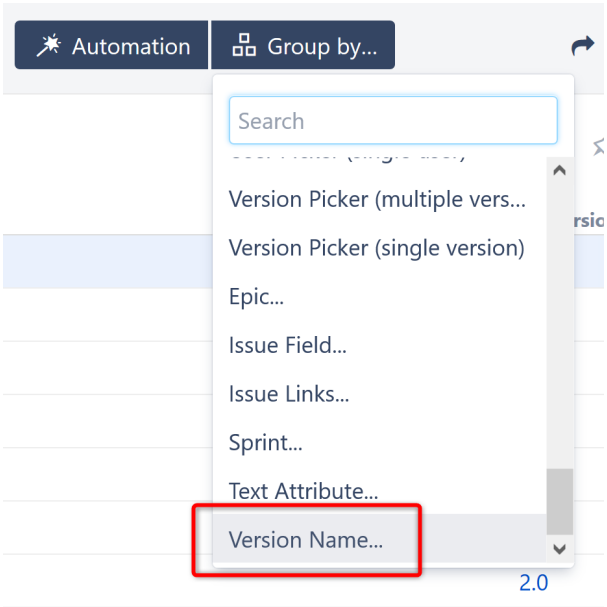
This can be very useful when trying to identify items that don't have a value for a particular field. For example, we can quickly see that "Team A Story 20" is not linked to an epic. To link it to an epic, we can simply drag it to the appropriate folder.

3.4.8.6 See Also

- Group by Version Name(see page 173)
- Group by Attribute(see page 175)

3.4.8.7 Group by Version Name

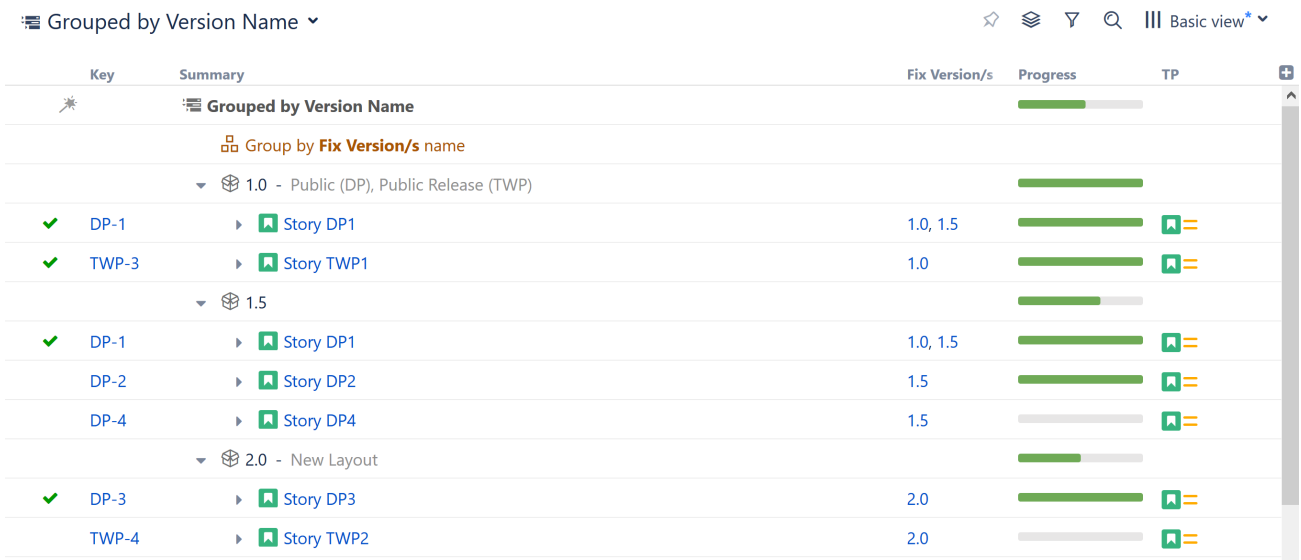
Using the **Version Name...** Grouper, you can group issues with the same version names across multiple projects.



You can select whether to group issues by the Affects Version, Fix Version or a version custom field.

Multiple Projects, Brought Together

When you use the **Version Name...** Grouper, issues with the same version name are grouped together, regardless of which project they appear in.



> The Version Name Grouper is case insensitive and ignores spaces, so "Version 1" and " version1" will be in the same group.

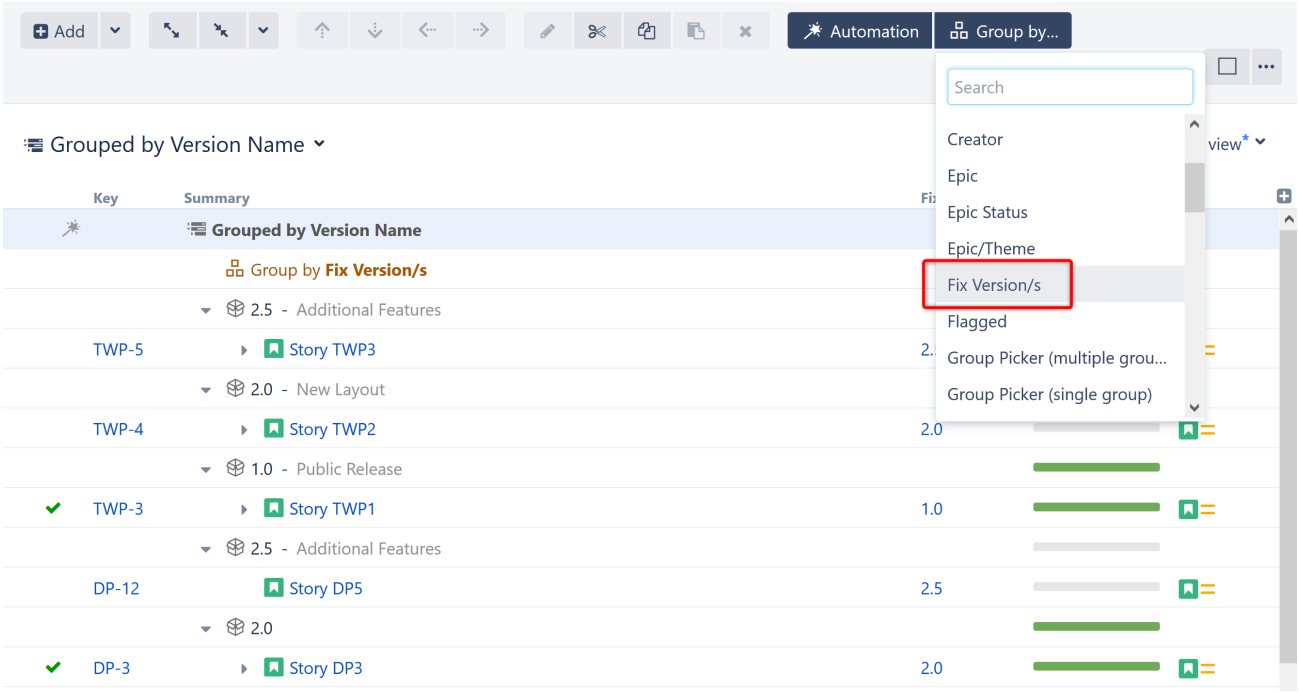
Version Descriptions

As you can see in the screenshot above, Version groups also include the version description, when available. Descriptions are listed in the following manner:

- If only one project includes a description for a particular version, or if every project has the same description, only the one description is displayed (see Version 2.0 above).
- If each project has a unique description for the version, all descriptions will be displayed with the corresponding project listed in parenthesis (see version 1.0 above).
- If no project includes a description for a version, no description is displayed (see version 1.5 above).

Grouping by the Attribute Rather Than Version Name

The **Version Name...** Grouper creates a single group for each version, regardless of which project the issues come from. If you prefer to have separate groups for separate projects, instead of selecting **Version Name...** from the Group by menu, select the specific version attribute you wish to group by (Affects Version/s, Fix Version/s or a version custom field).

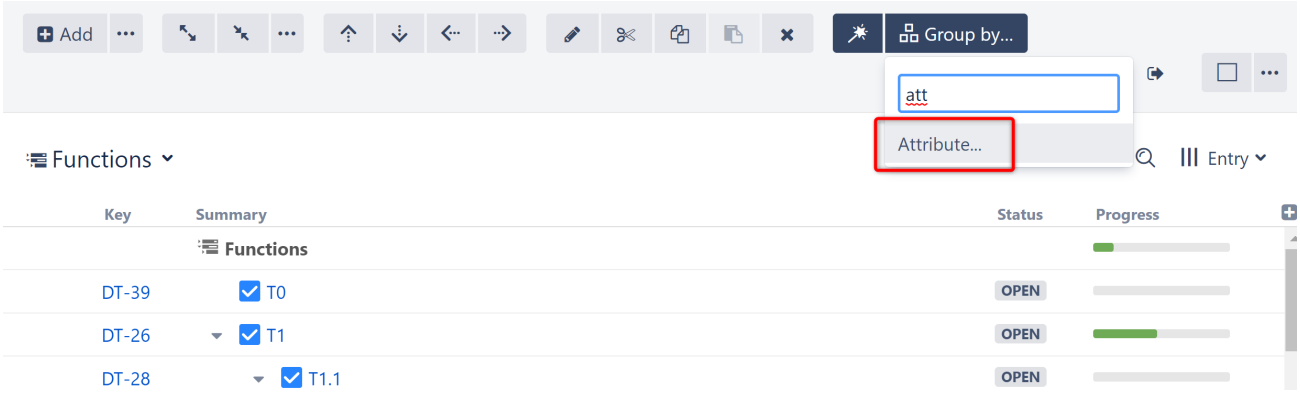


Notice that when grouping by the attribute itself (rather than the Version Name Grouper), you may wind up with multiple groups per version, because versions are not combined across projects.

3.4.8.8 Group by Attribute

The **Attribute...** option allows you to group items by read-only values and non-Jira fields, such as:

- Formulas
- Dates
- Structure attributes
- Item properties
- Structure extension attributes (Structure.Testy statuses, etc.)
- Tempo Teams



Attributes are read-only, so you cannot drag issues between groups to make changes, as you can with other Group generators(see page 169).

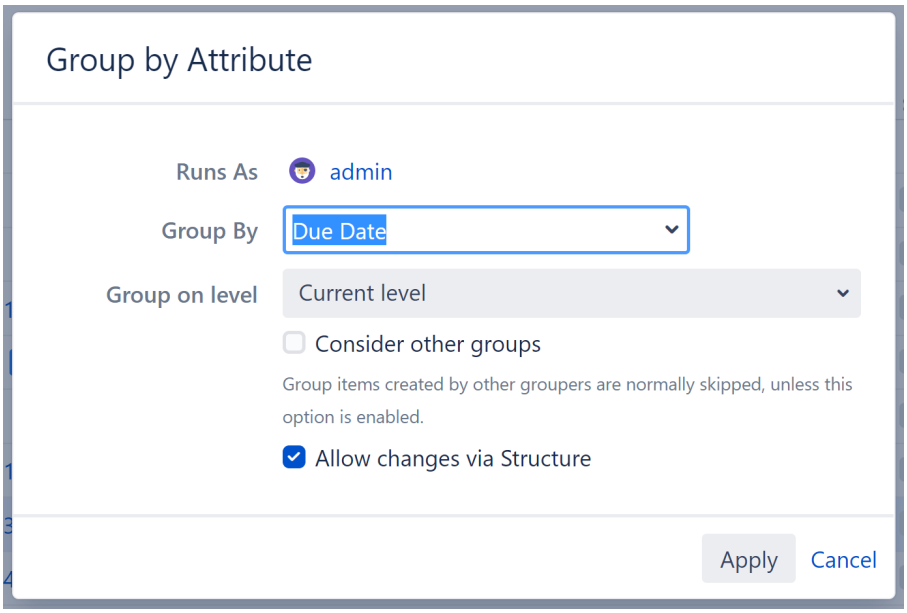
Item Values and Multi-Valued Attributes

If the attributed selected, for example, a formula, provides a multi-valued result (represented by an array in the formula), the generator will create multiple groups for the item – similar to what you'd expect when grouping by Fix Versions, for example.

If the value provided by the attribute is a text, then each group will be represented by a folder. If the value is an item – a user, a priority, another issue, or a similar object – the generator will use those items as groups.

Adding a Group by Attribute generator

To group issues by an Attribute, first select **Automation | Group by... | Attribute...** in the Structure toolbar. On the settings screen select the attribute you want to use.



Then select the following options:

- **Group on level** - You can apply the grouping to the current level, next level, or manually enter a level within the hierarchy. See Generator Scope(see page 198) to learn more.
- **Consider other groups** - By default, a new Group generator will ignore data created by other Group generators.
- **Allow changes via Structure** - Please ignore this. Attributes are read-only, so it is not possible to make changes via Structure. We plan to remove this option in a future release.

3.4.9 Sort Generators

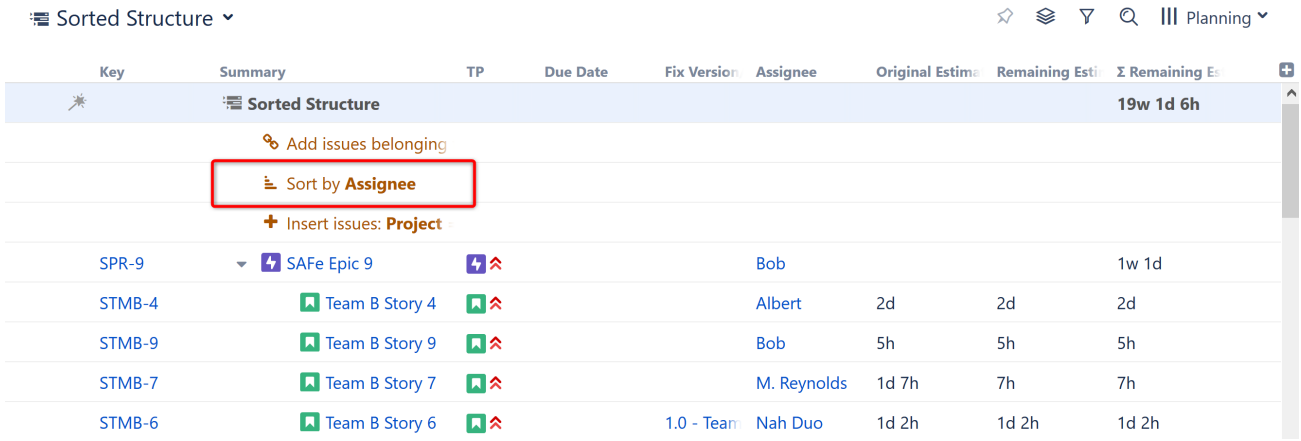
Sort generators allow you to order your structure based on a Jira attribute, Structure attribute, or Agile rank. They can also be used to allow manual sorting for JQL-generated structures.

While sorting is possible from within the structure itself (by clicking the row you want to sort by), the Sort generator allows you to fully customize the ordering of items within your hierarchy - the top level can be sorted by one attribute, while lower levels are sorted by another. Or you can add several different sorts using a manual level range.

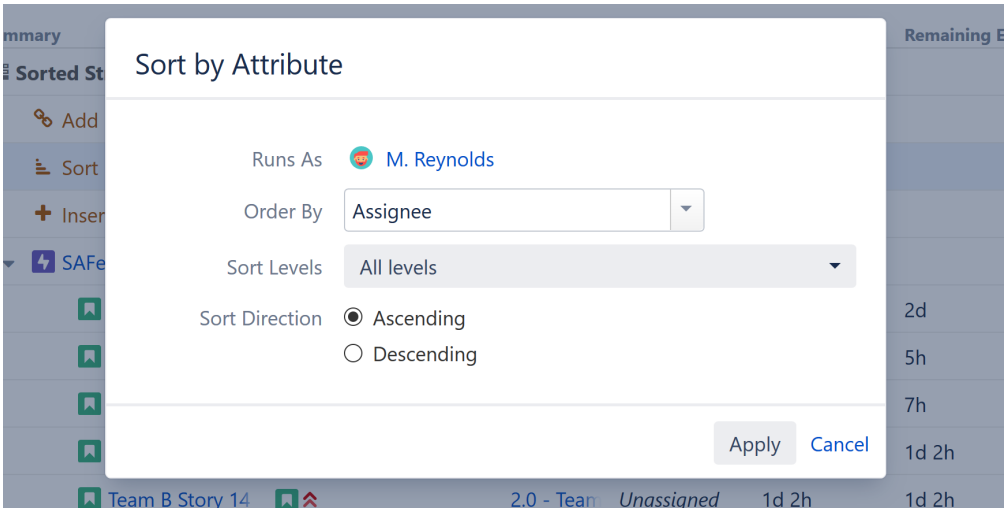
Placement matters. Generators only affect issues beneath them, so if you want to sort the entire structure, place the generator at the very top (by selecting the structure's name in the top row). If you place it anywhere else, it will only sort the items beneath it.

3.4.9.1 Customize Your Sort

Most of the options under the "Sort By..." Automation are applied the moment they are selected, and you will not be asked to set parameters for the sort. However, you can still customize a Sort generator by locating it within your structure and double-clicking its summary.



From the **Sort by Attribute** dialogue, you can select a new attribute to sort your structure by, change the level(s) within your hierarchy where the sort is applied, and/or change the order of the sort.



Sort generators can be applied to all levels, the current level only, or to a custom level range.

To customize these values right away, in the **Automation | Sort by...** menu select **Attribute**.

3.4.9.2 Level-based Sorting

If you want different levels within your hierarchy sorted by unique parameters, simply add multiple Sort generators, each with different Manual Sort Levels.

For example, if you need to quickly assess the progress of sprints across your organization, you could Insert (see page 142)issues from all active projects (and use Extenders(see page 148) as appropriate), and then apply the following sorts:

- Sort by Project - Applied to the Top Level
- Sort by Sprint - Applied to the Second Level
- Sort by Progress - Applied to the Third Level

See Generator Scope(see page 198) to learn more about customizing levels.

> If your hierarchy has some levels you don't want sorted, or multiple levels you want sorted by the same attribute, that's okay - just set a specific range for each type of sort.

3.4.9.3 Rank Sort

If you use Sort by Rank, Structure can update the Rank as a user moves issues up and down inside the structure.

3.4.9.4 Manual Reordering Generator

If your structure was built using a JQL Query Inserter(see page 143), by default you will not be able to reorder issues within the structure. The **Manual Reordering Generator** allows you to move issues up or down, provided they remain at the same level within the hierarchy.

> To move items between levels, you can enable Manual Adjustments(see page 194).

3.4.9.5 Additional Information

- Sort Order(see page 178)

3.4.9.6 Sort Order

When a structure is sorted, either by a Sort Generator(see page 178) or Sort Transformation(see page 216), the following order is used:

- **Numeric values** - are sorted in numeric order
- **Text values** - are sorted alphabetically
- **Dates** - are sorted chronologically
- **Priorities** - are sorted based on the Jira Priority order[33]
- **Sprints** - are sorted first by boards, and then alphanumerically by the sprint summary
- **Statuses** - are sorted first by status category, and then by their position on the Statuses page[34] in Jira Administration

33 https://confluence.atlassian.com/adminjiraserver/defining-priority-field-values-938847101.html
34 https://confluence.atlassian.com/jirakb/add-new-status-and-steps-to-jira-server-workflows-718835875.html

- **Versions** - are sorted first by project, and then by the position of the current version on the project version page

3.4.10 Effectors

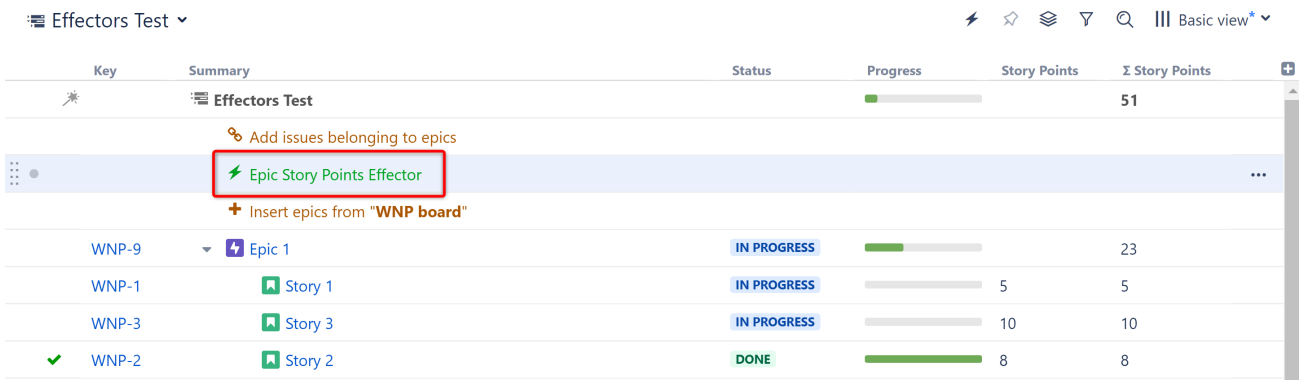
Effectors allow users to update Jira fields with information only available in Structure.

Sorry, the widget is not supported in this export.
But you can reach it using the following URL:

https://www.youtube.com/watch?v=QSreuWUQTp8

3.4.10.1 Attribute to Issue Field Effector

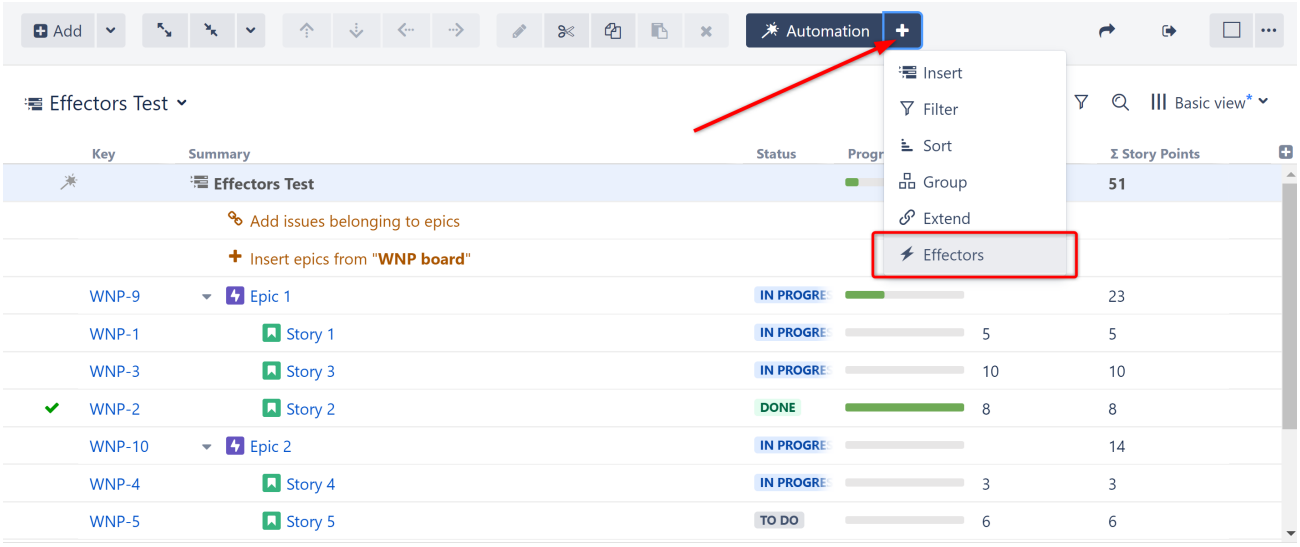
The Attribute to Issue Field Effector allows you to write the values from Structure attributes (formulas, structure-specific columns, Structure.Gantt attributes, etc.) to Jira issue fields. It can also copy values from one Jira field to another.



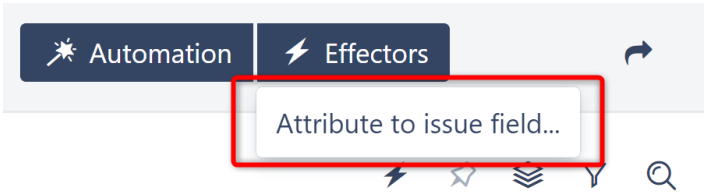
In the above example, we've added an Effector that will pull the aggregate value from the Structure-specific ∑ Story Points column and write it to each Epic's Story Points field. You can see exactly how we set it up in our Calculate Epic Story Points Based on Sub-issues(see page 637) article.

Adding an Attribute to Issue Field Effector to a Structure

To add an Attribute to Issue Field Effector to a structure, open the Automation menu and select **Effectors**.



Choose **Attribute to issue field...**



On the Effector setting screen:

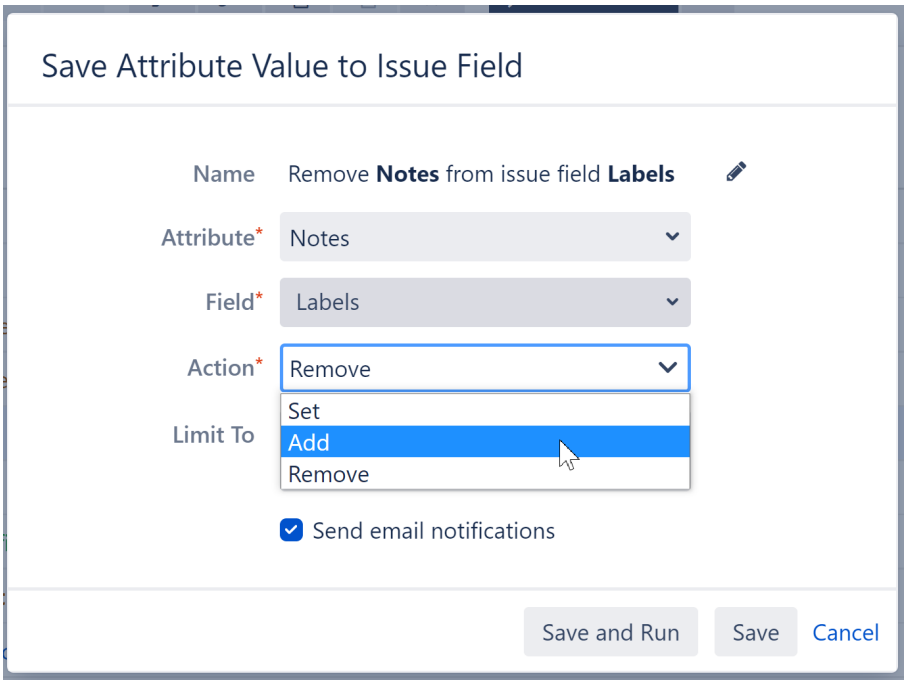
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Attribute: select the source attribute (with the values you want to write to Jira)
- Field: select the field you want the values written to
- Limit To: (optional) enter JQL to limit the Effector's scope (Ex. "Issuetype = epics" will limit changes to just epics)
- Select whether email notifications should be sent when the Effector writes values to Jira

When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet. Since this will be your first time running an Effector, we suggest clicking **Save**. This way, you can run a pre-test before making any changes.

Writing to Multi-Value Fields

If you select a destination Field that can accept multiple values, you will see an additional Action setting. You can choose:

- Set - the values in the issue field will be overwritten by the attribute value.
- Add - the value from the source Attribute will be added to the values in the destination Field.
- Remove - if the value in the source Attribute appears in the destination Field, it will be removed.



> ⚠ Currently, only single values can be written to the issue field. If the source Attribute contains multiple values or spaces (as with the Notes field), you will receive an error when you run the Effector.

Running the Effector

Once you've added the Effector to a structure, you need to run it for changes to take place. See Running an Effector(see page 186)

Review / Revert Changes

To review or revert changes made by an Effector, see Revert Effector Changes(see page 190)

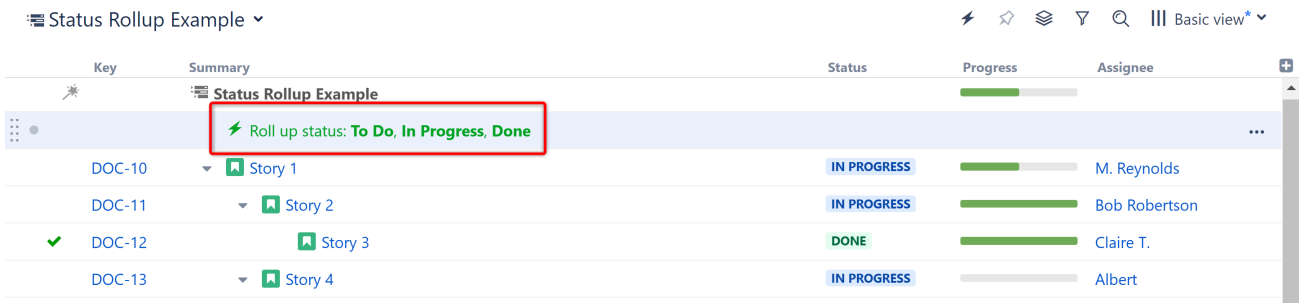
Example Use Cases for the Attribute to Issue Field Effector

Some popular uses for this Effector include:

- Add story points for epics(see page 637)
- Update issue assignees based on parent issues(see page 664)
- Calculate the cost to complete issues and projects, and write those cost to a Jira field(see page 642)
- If the Structure.Gantt extension is installed, write Gantt attributes (such as the Gantt Start Date/Gantt Finish Date) to custom Jira fields (Start Date/Finish Date).

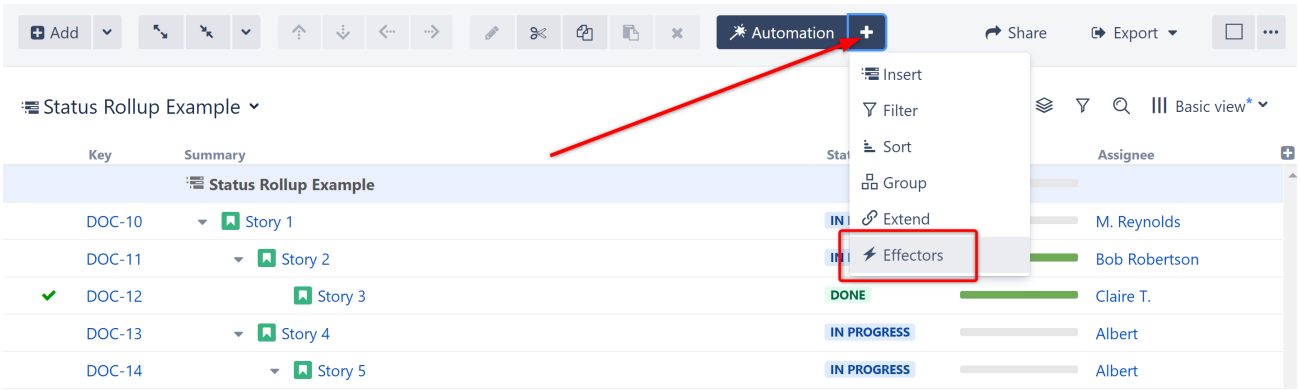
3.4.10.2 Status Rollup Effector

The Status Rollup Effector automatically updates parent issue statuses based on the earliest status of their sub-issues. For example, if all sub-issues are *Resolved,* it will update the parent issue's status to *Resolved*; but if one sub-issue is still *In Progress*, the parent issue will be set to *In Progress.*

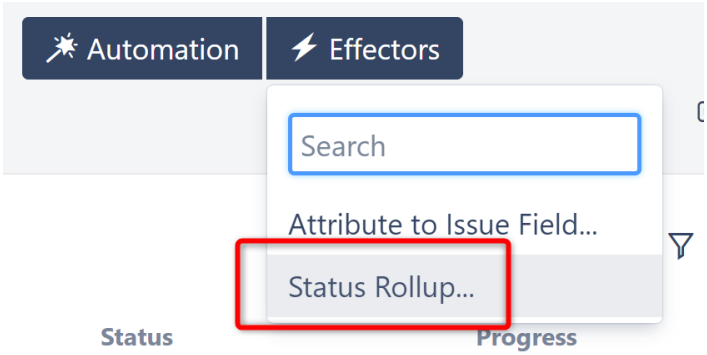


Adding a Status Rollup Effector to a Structure

To add a Status Rollup Effector to a structure, open the Automation menu and select **Effectors**.

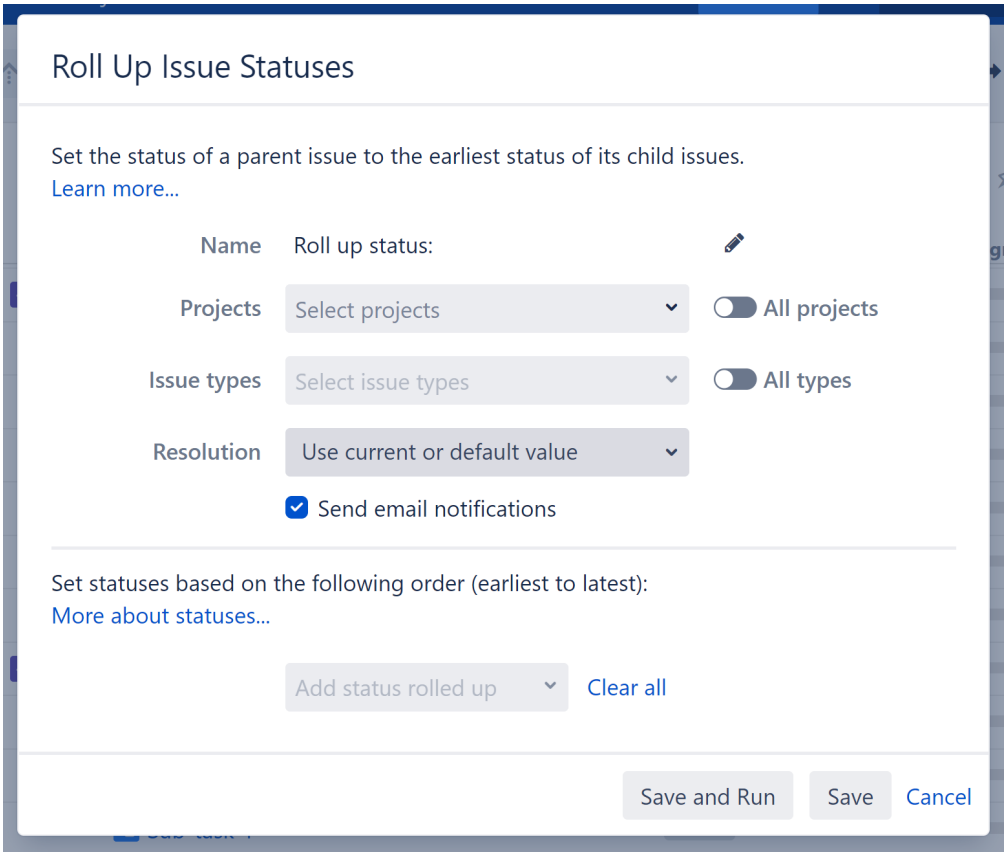


Choose **Status Rollup...**



On the Effector setting screen:

- **Name:** this field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- **Projects:** select the projects that should be updated by the rollup. To include all projects, toggle **All projects**.
- **Issue types:** select which issue types should be considered by the rollup. To include all statuses, toggle **All types**. *Note: If your structure contains issue types not included here, the Effector will not make any changes to those issues OR their ancestors (items above them in the structure).*
- **Resolution:** this resolution will be used whenever an issue is moved to a "resolved" state (based on the project's workflow).
- Select whether email notifications should be sent when the Effector writes values to Jira
- **Set statuses based on the following order:** select the statuses that should be considered by the Effector, and place them in order from earliest status to latest. If you do not want the Effector to move issues to a specific status, do not include it here. For example, leaving out "Done" will prevent issues from being closed when all their sub-issues are Done.



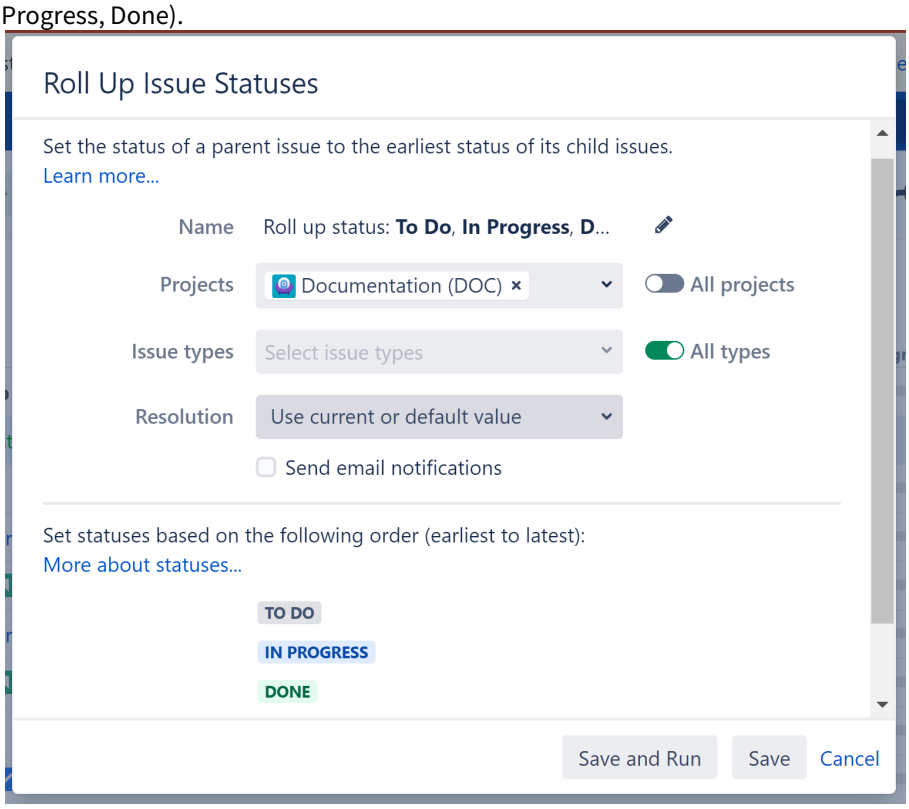
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

How the Status Rollup Effector Works

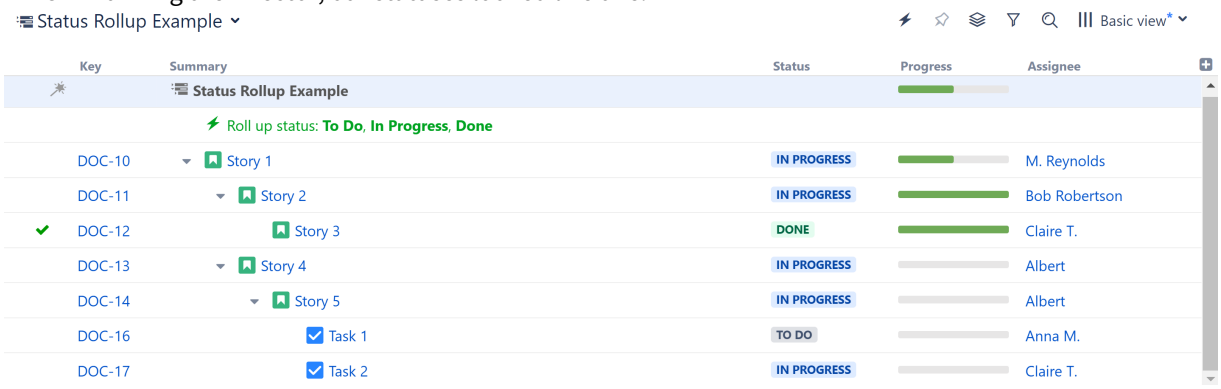
When you run the Effector, it updates the status of parent issues to the earliest status of their sub-issues, based on the project(s), issue type(s), and status order you chose in the configuration.

Let's take a closer look at how this works.

1. We configured our Effector to our DOC project, selected ALL issue types, and set our statuses (To Do, In Progress, Done).

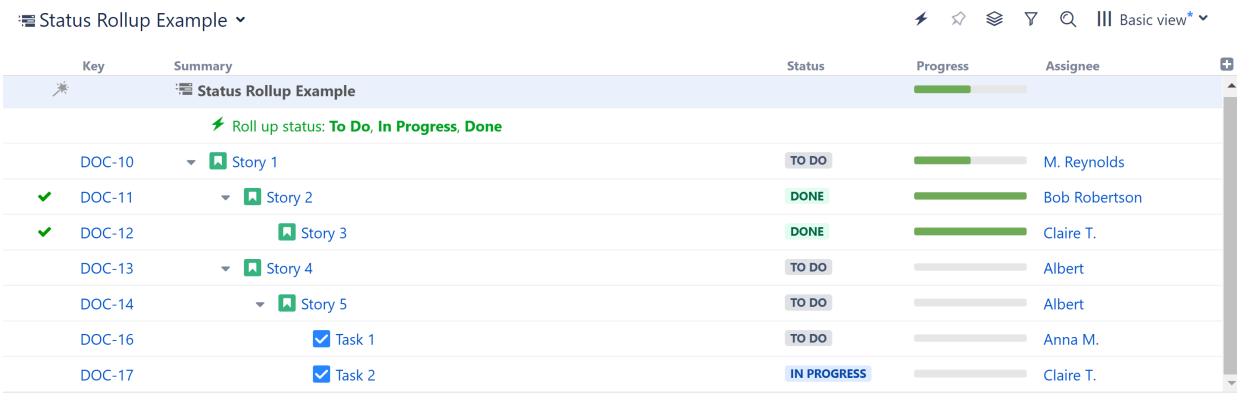


2. BEFORE running the Effector, our statuses looked like this:



3. AFTER running the Effector, the following status changes were made:

   - **Story 2** was moved to DONE, because it's only sub-issue, **Story 3**, was DONE.
   - **Story 1**, **Story 4** and **Story 5** all share a sub-issue (**Task 1**) with the earliest possible status (TO DO), so they were all moved to TO DO.



Missing Status

Statuses that are not selected in the settings are not recognized by the Effector. If a sub-issue has one of the unselected statuses, it will not affect it's parent issue. Additionally, if a parent issue has an unselected status, it will not be updated by the effector, even if one of its children has an earlier status.

This can be useful if you want to prevent issues from being transitioned to or from certain statuses. For example, if you omit DONE from your list of statuses, no issue can be moved to or taken out of DONE status by the Effector.

Missing Issue Type

If an issue type is not selected in the settings, the Effector will not change the issue's status or the status of any of its ancestors (issues above it in the structure).

For example, in our example above, if we had not included Tasks in our list of issue types, the Effector would have ignored Task 1 and Task 2, and it would not have changed the statuses of any of their ancestors (Issues 5, 4, and 1).

Running the Effector

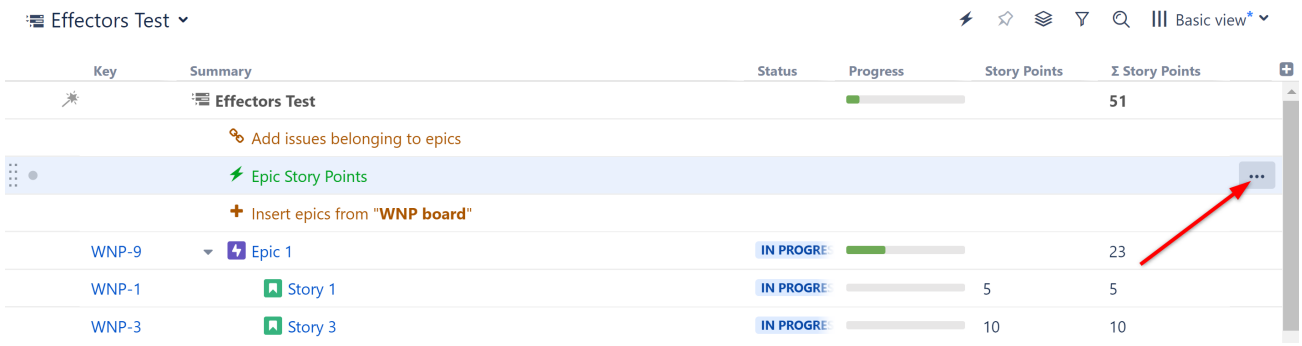
Once you've added the Effector to a structure, you need to run it for changes to take place. See Running an Effector(see page 186)

Review / Revert Changes

To review or revert changes made by an Effector, see Revert Effector Changes(see page 190)

3.4.10.3 Running an Effector

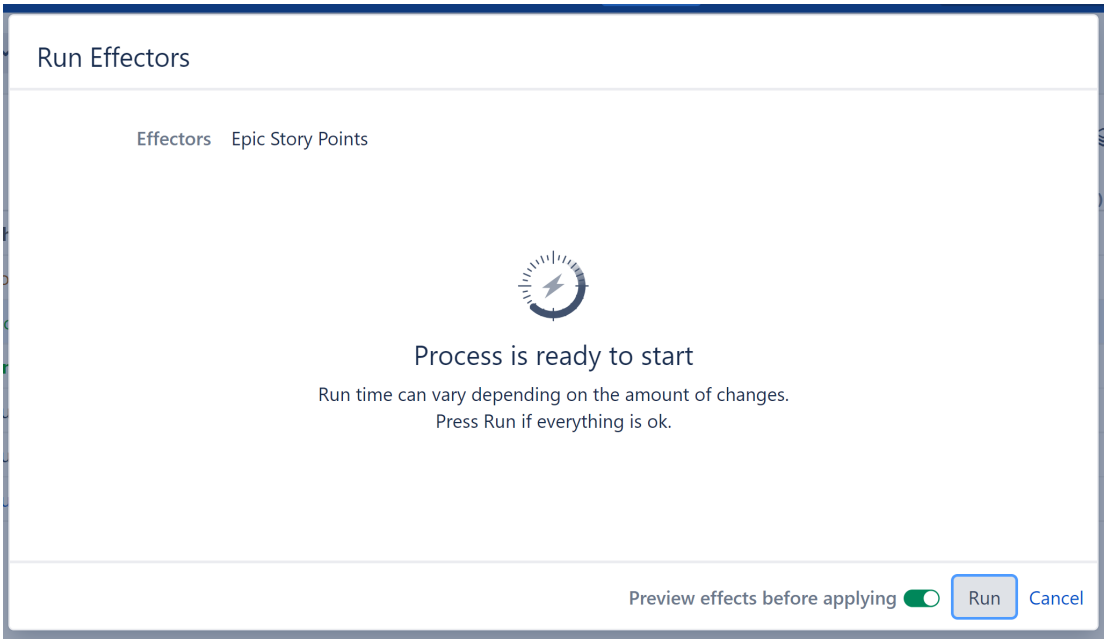
Once you create an Effector, you need to run it for any changes to take effect. To do that, locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.



In order to run an Effector, user must have Bulk Change global permission in Jira.

Preview Option

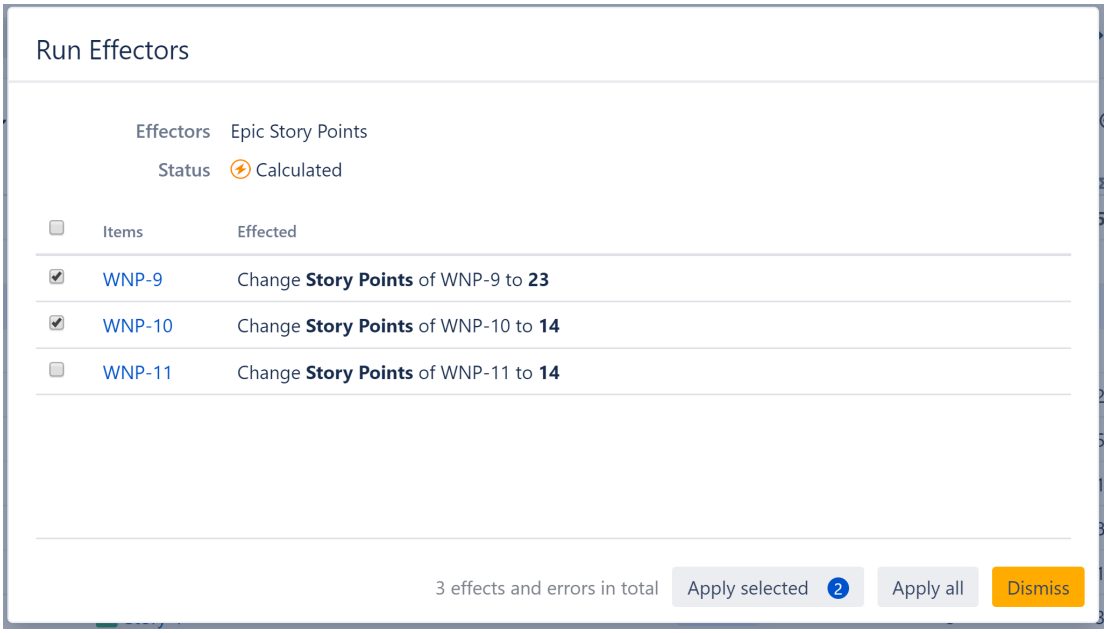
By default, Effectors initially do a Preview run, which allows you to view and approve every change that will be made by the Effector, without making any changes to Jira data. If you prefer to simply apply changes without reviewing them, switch the **Preview effects before applying** toggle off. When you're ready to run the Effector (in Preview mode or not), click **Run** to begin.



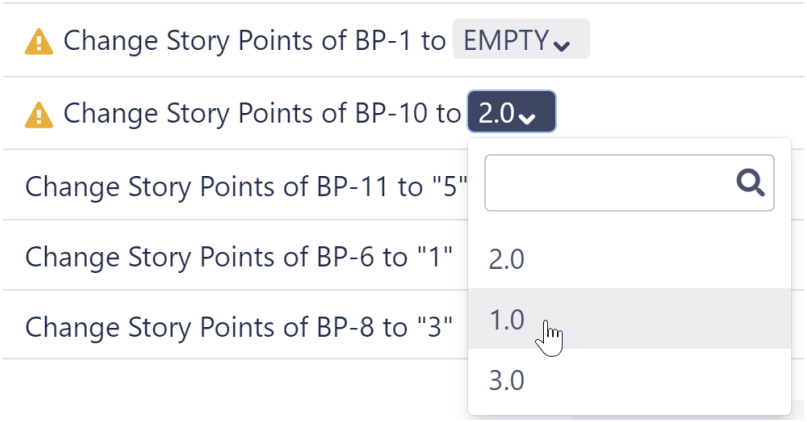
Effectors update live Jira data. We highly recommend using the preview option.

Preview Changes

If you selected Preview effects before applying, once the preview finishes you will see a list of changes that will be made by the Effector. You can select specific changes you want made, or click **Select All** to apply all changes.



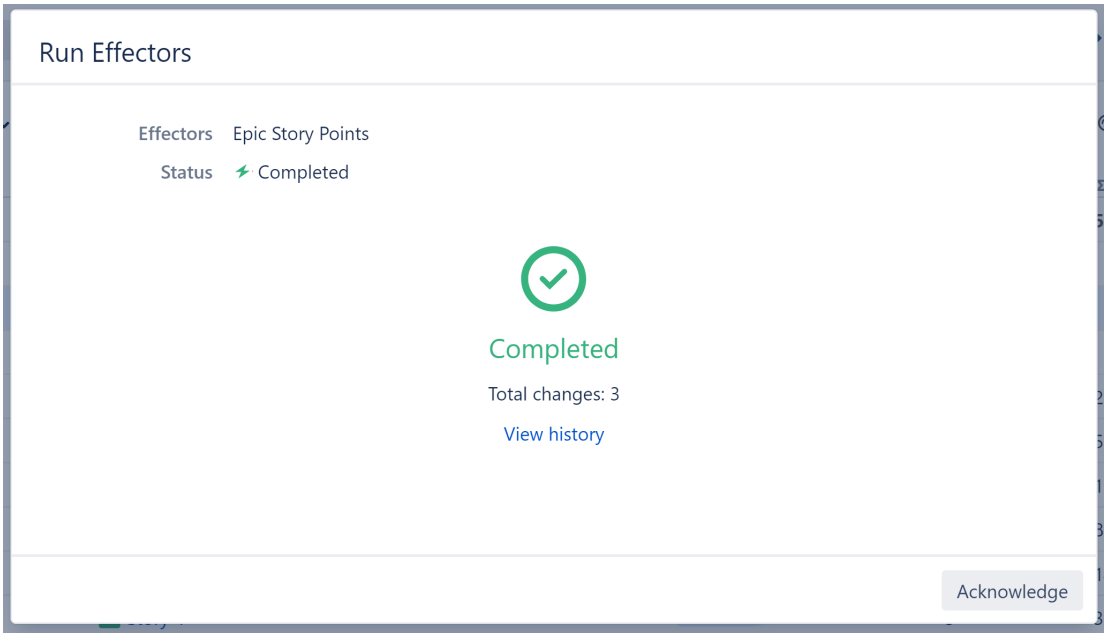
If an issue has more than one value within the structure, you will have a chance to select the value or values you want to write to the Jira field.



Once you click **Apply selected** or **Apply**, the Effector will begin processing your changes and updating values in Jira.

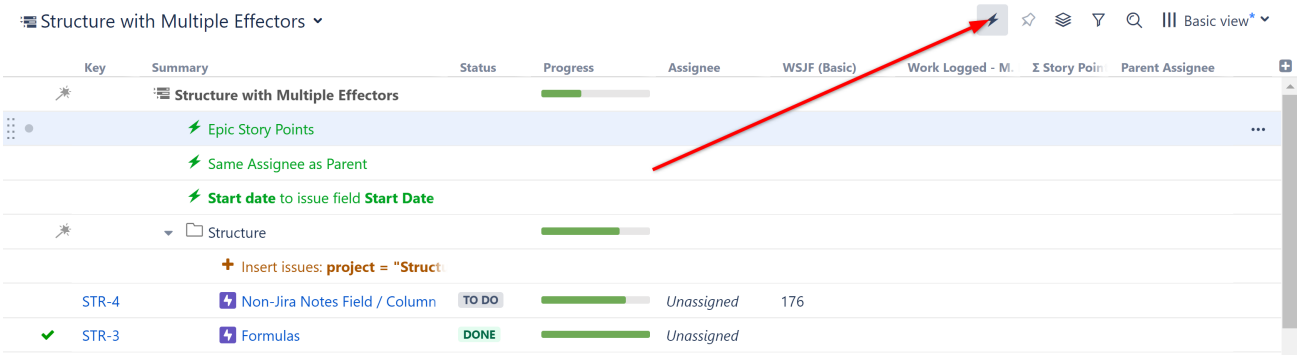
> If you click **Dismiss**, no changes will be made to Jira and the preview will be discarded. To apply changes later, you will need to re-run the Effector.

Once complete, you will see a confirmation letting you know that all changes were made. If the Effector is unable to make some or all changes, that information will be displayed here as well.

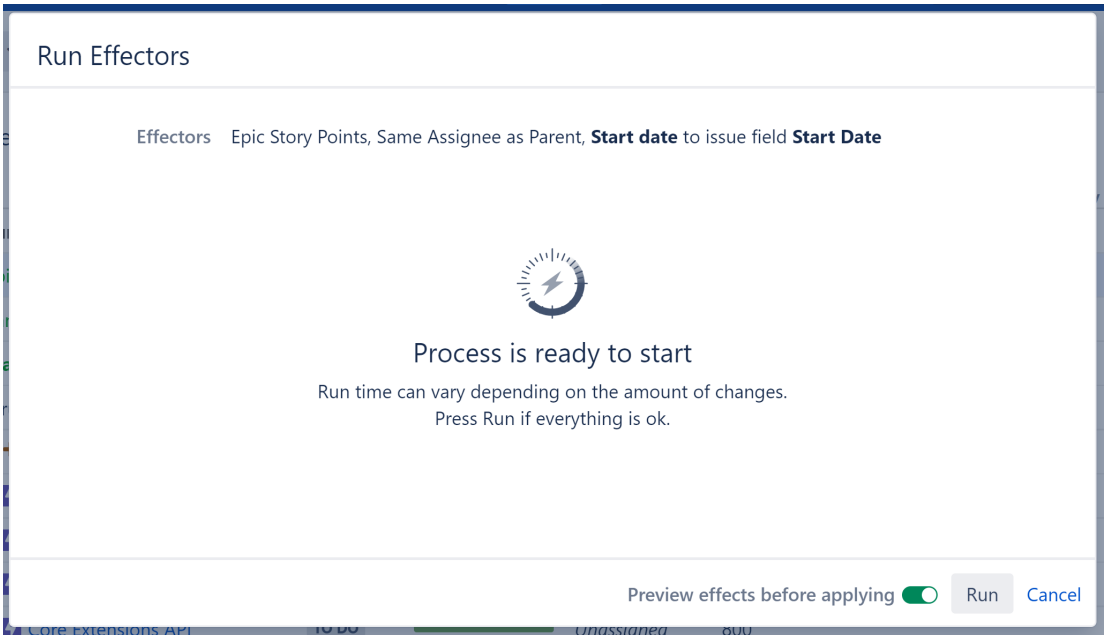


Running All Effectors in a Structure

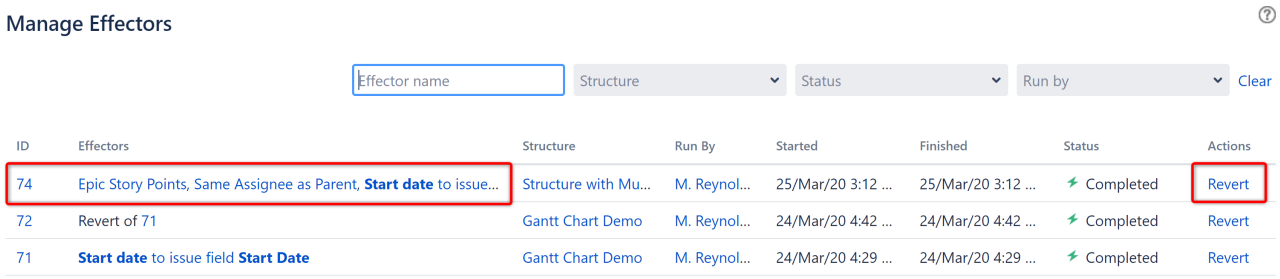
If you have multiple Effectors in a structure, you can run all of them at once by clicking the **Run all effectors in this structure** button:



Just as when you run a single Effector, you can select the "preview" mode to review any changes before they are applied to Jira.



Once the changes have been applied, you can review the operation on the Manage Effectors(see page 193) screen and, if necessary, revert any changes(see page 190).



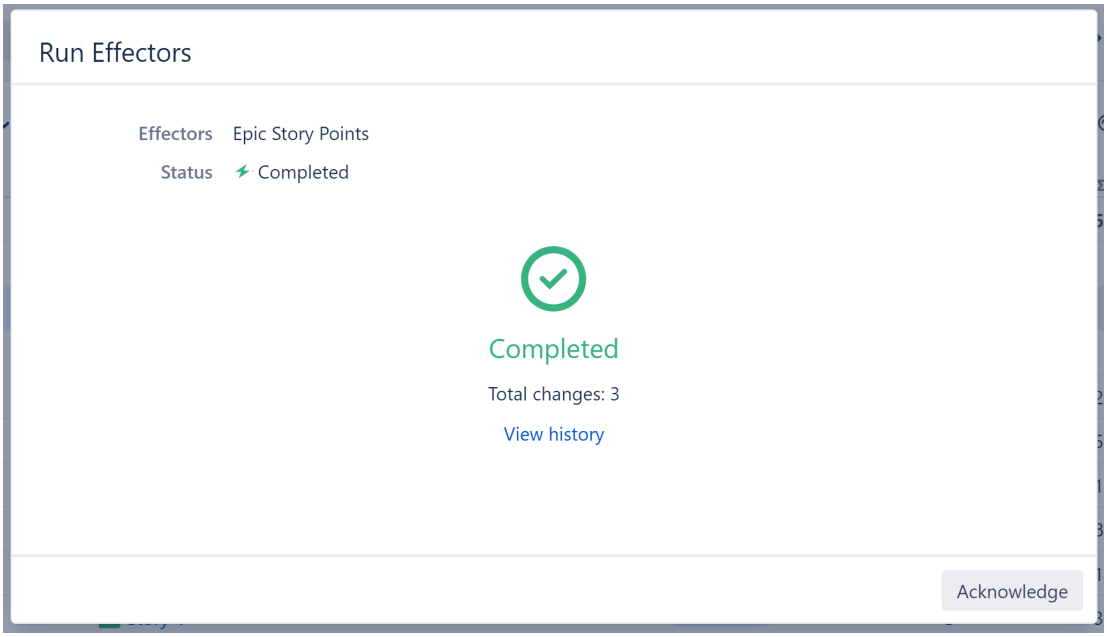
3.4.10.4 Revert Effector Changes

You can revert changes made by an Effector:

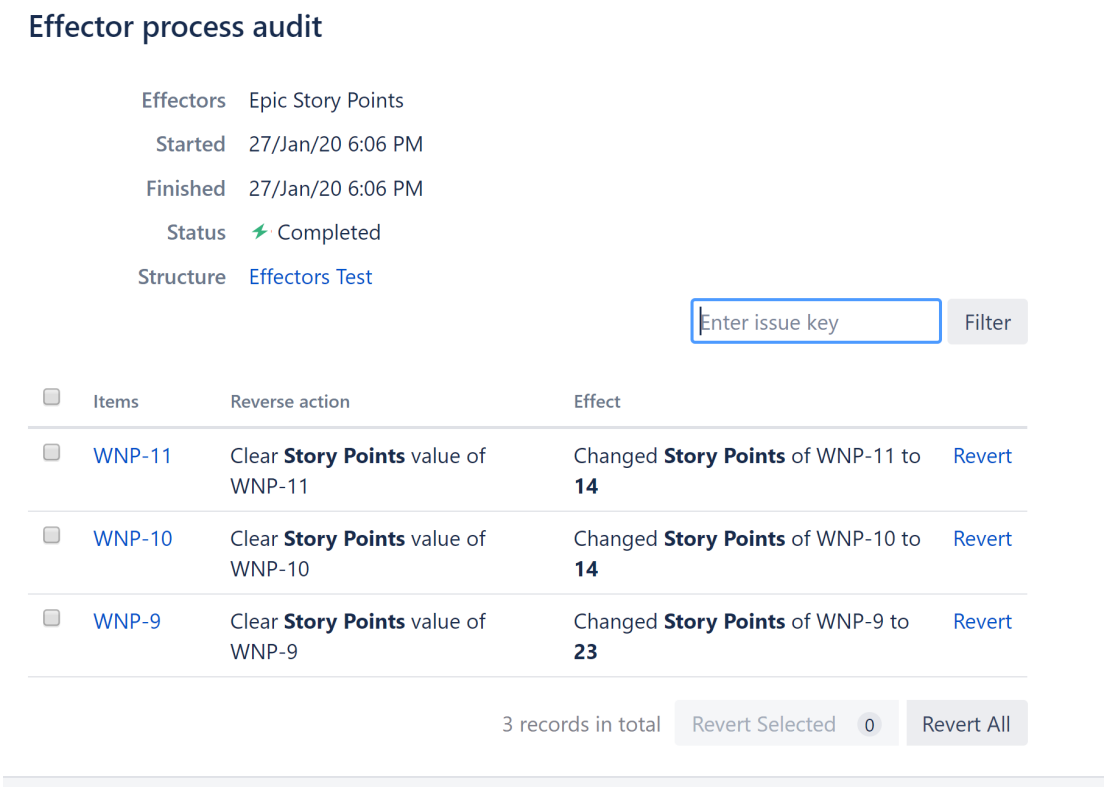
- Immedatiately after running the Effector(see page 190)
- At any time, using the Manage Effectors page(see page 192)
- You can even revert changes made during a revert process(see page 192)

Revert Effector Changes Immediately After an Effector is Run

Once an Effector finishes applying changes, you will see a confirmation letting you know that all changes were made. If the Effector is unable to make some or all changes, that information will be displayed here as well.

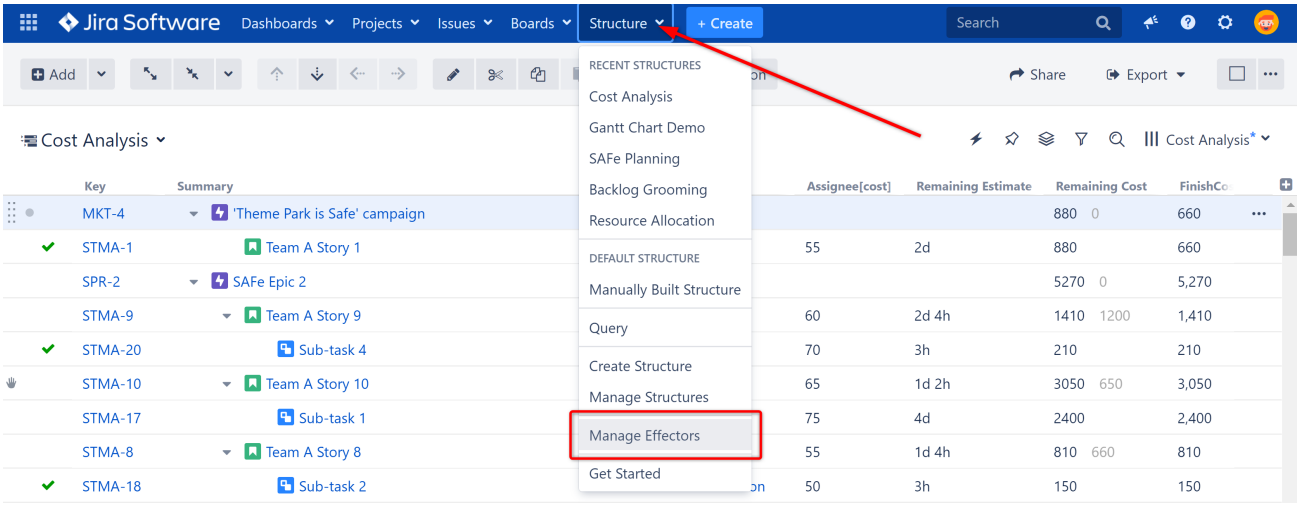


To revert any of these changes, click the **View history** link to open the Effector Process Audit screen. From there, locate the change you need to revert and click its **Revert** link. To revert multiple changes, select each and click **Revert Selected**, or click **Revert All**.

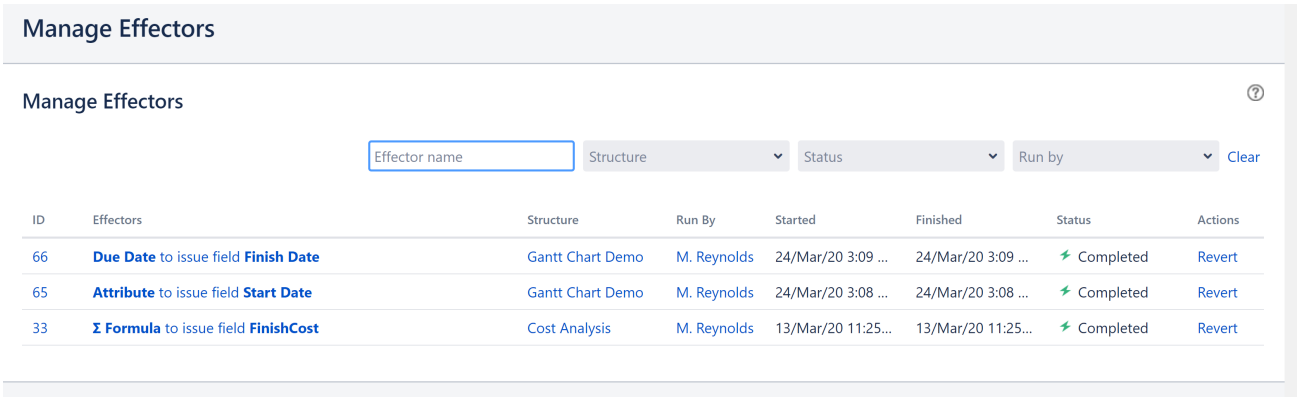


Revert Effector Changes using the Manage Effectors Page

You can review and/or revert changes made by Effectors at anytime. Simply go to the Structure menu and select **Manage Effectors**.



The Manage Effectors page lists every time an Effector has been run.



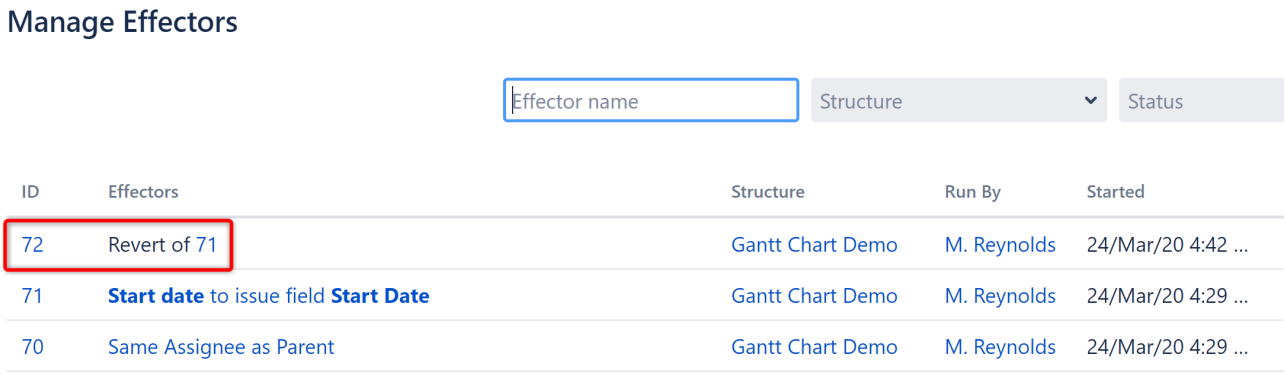
From here, you can review and revert changes from these Effectors.

- To revert all changes made by an Effector's run, click the **Revert** link next to the appropriate run.
- To review each change and select only certain changes to revert, click the ID for the run.

> ⚠ In either case, this will revert the current field values to the values prior to this Effector's run, even if other changes have been made since.

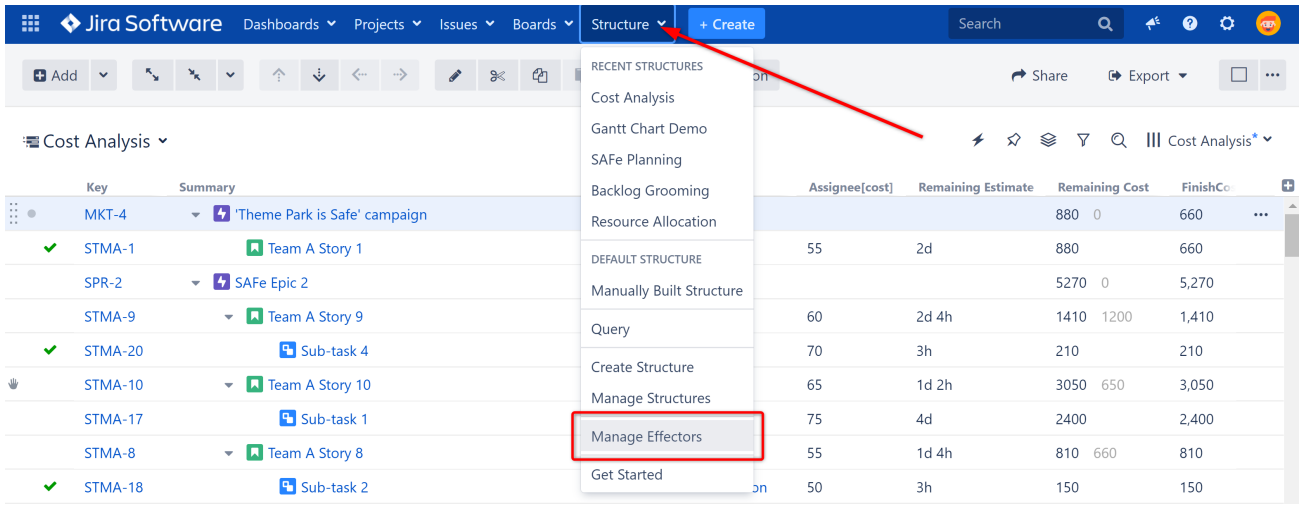
Revert Changes Made During a Revert Process

After you revert changes from an Effector, a new process will be added to the Manage Effectors screen. You can review and revert changes from this process just as you can with any other.



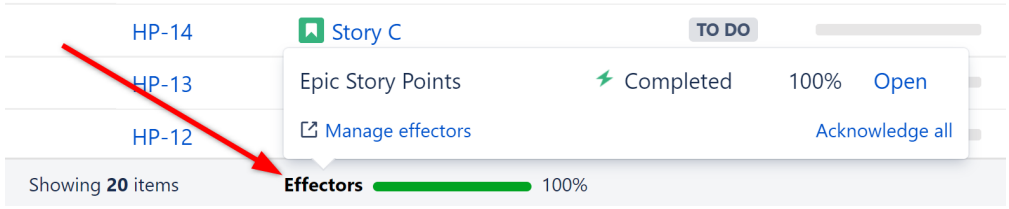
3.4.10.5 Managing Effectors

You can review all the Effectors you've run as well as the changes they've made at anytime. Simply go to the Structure menu and select **Manage Effectors**.



Jira admins can review all Effector processes on the Jira instance. All other users will only see their own.

You can also review Effectors you've run by clicking the Effectors link in the Structure Board footer.

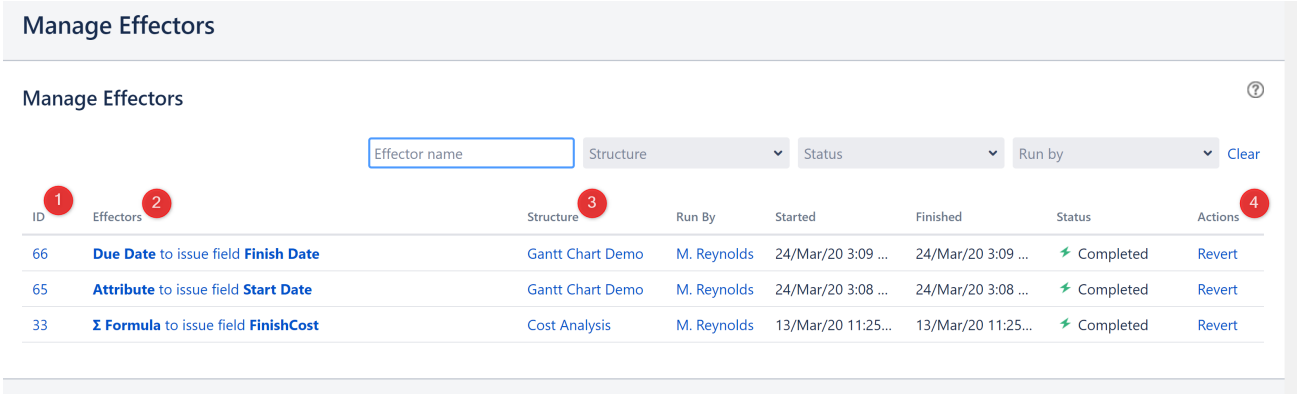


To view Effectors for a specific structure, open the Manage Structures page, locate the structure and click the number in the Effectors column.

Manage Effectors Page

The Manage Effectors page lists every time an Effector has been run and allows you to take several actions:

1. Click the ID to view all changes made when an Effector was run and revert(see page 190) select changes if needed.
2. Click the Effector's name to view the Effector in its structure.
3. Click the structure name to view the structure containing the Effector.
4. Click Revert (see page 190)to undo all changes made when the Effector was run. Note: this will revert the current field values to the values prior to this Effector's run, even if other changes have been made since.



Finding an Effector Process

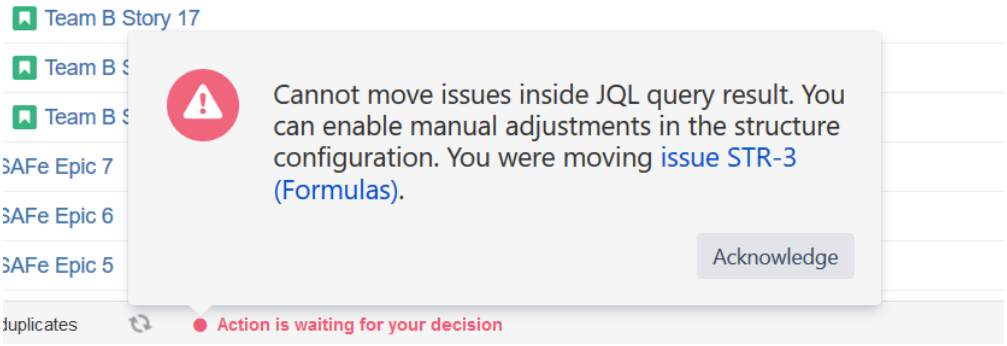
To locate a specific Effector process, enter the Effector name (or as much of it as you can remember) into the search box at the top of the screen. If you still can't find it, you can filter the results by:

- Structure, including Structure ID
- Status of the Effector process
- Who ran the Effector process

3.4.11 Manual Adjustments

Manual Adjustments allow you to move dynamic content within a structure, regardless of the Automation used to create the structure.

If you used Generators(see page 140) to build all or part of a structure, the content it adds to your structure cannot be moved as freely as content which has been manually added to a structure. This is because Automation uses generators to dynamically add content from Jira, and then continuously checks that content against Jira to keep both up-to-date. If you attempt to move an item in your structure in a way that does not fit within your generators' rules, you will receive an error message.



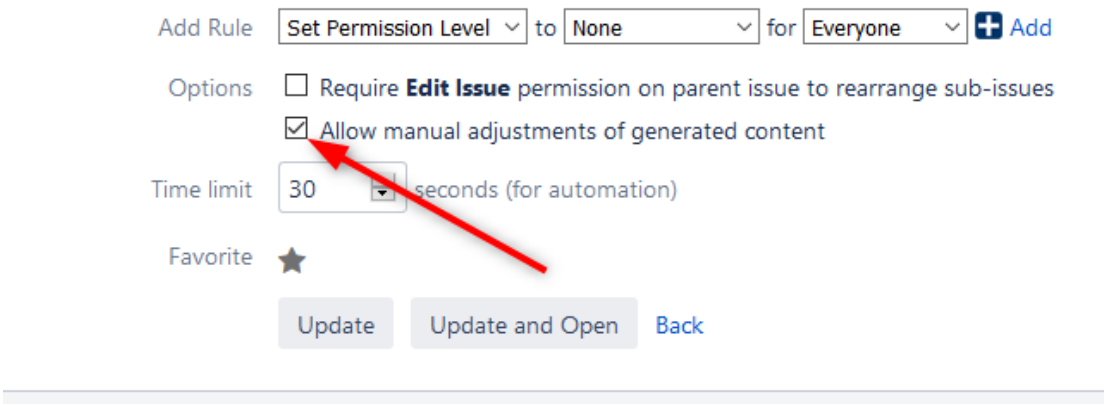
There may be times, however, when you need to move those dynamic items around freely, regardless of your generator rules. To do so, you need to enable Manual Adjustments.

3.4.11.1 Enabling Manual Adjustments

When Manual Adjustments are enabled, you can move items anywhere within your structure, regardless of the generators used to create it. For example, you can drag items into a custom folder, move tasks under a different project, or create your own custom hierarchy – just as you can if you create a structure without automation.

To enable Manual Adjustments:

1. Go to **Structure** in the top menu and select **Manage Structures**
2. Select the structure you wish to update, and under Options select **Allow manual adjustments of generated content**



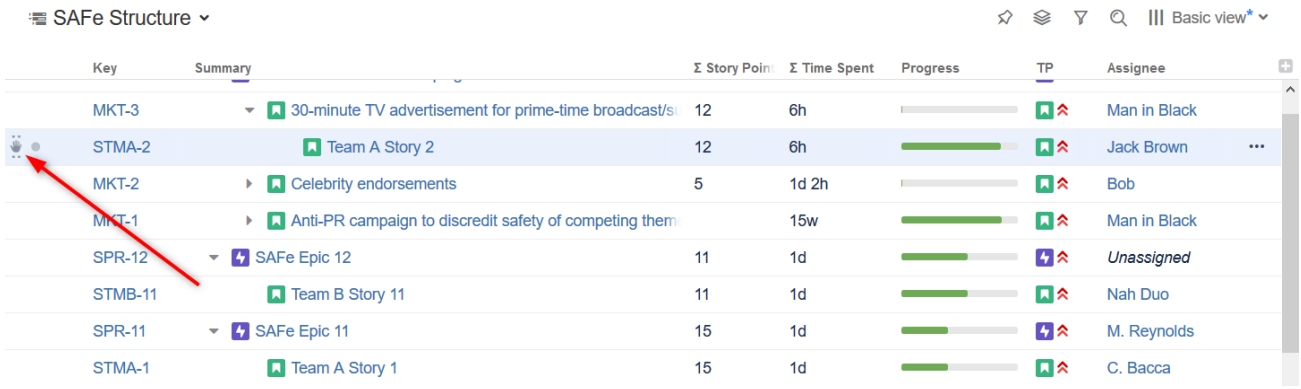
If you cannot enable/disable manual adjustments, you may not have the appropriate permissions. Speak to your Jira administrator.

Once you have enabled Manual Adjustments, any move you make will continue to be checked against your existing generators. However, now you can have two different outcomes:

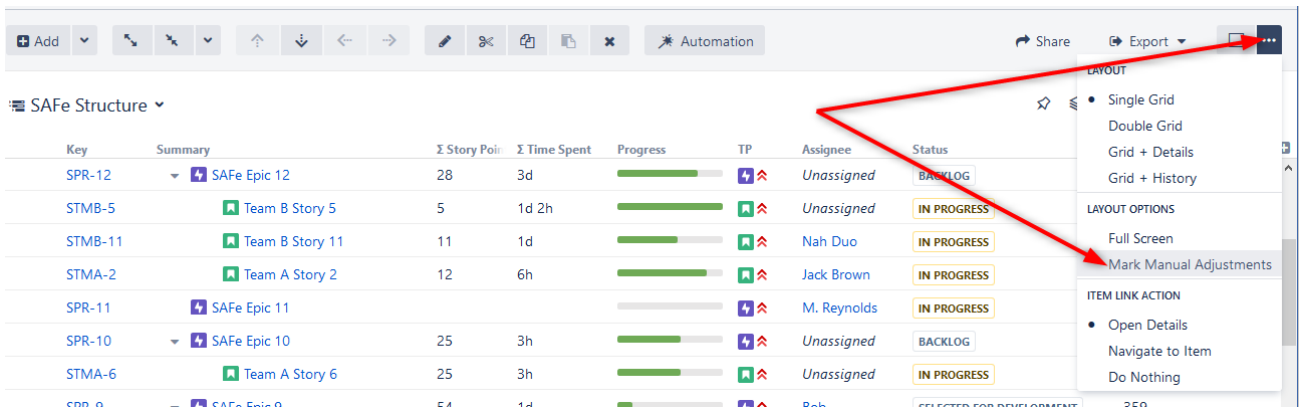
- If the move fits within the generators' rules, your content will be moved just as it was previously, and that move will be reflected in Jira.
- If the move does not fit with the generators' rules, your content will be moved using Manual Adjustments, but that move will not be reflected in Jira.

Mark Adjusted Content

Any manually adjusted content within your structure will be marked with the Manual Adjustment icon.



You can hide the Manual Adjustment icons by deselecting the **Mark Manual Adjustments** in the **Toggle Panels** menu.



3.4.11.2 Manual Adjustments are NOT Reflected in Jira

When Manual Adjustments is enabled, some changes may not be reflected in Jira. This is because those changes do not fit within the rules of the generator(s) you are using. You can move items all you want for the purposes of your structure, but they will remain in their original location within Jira. If you created a new structure with the exact same generators, those issues would appear just as they had before you made manual adjustments.

3.4.11.3 Special Considerations When Using Manual Adjustments

Because Manual Adjustments are applied after Automation, certain types of moves may have different results than moving items within a manually-created structure. Please be aware of the following situations that may arise when using Manual Adjustments.

- **Moving Grouped Content -** If your structure is built using the Insert and Group generators, and you move all the issues out of a group, that group (now empty) will remain within your structure. This happens because the Group generator is run before the Manual Adjustment, so the folder remains in place, even though the issues were moved.

- **Moving Extended Content -** If your structure is built using the Insert and Extend generators, and you move one issue under another, no link will be created between the two issues. To create a link between the two issues, copy the original issue to the new location by holding the **ctrl** key (**alt** on Mac) while dragging the issue.

For more information, see Order of Operations with Manual Adjustments(see page 197)

3.4.11.4 Why is Manual Adjustment Necessary?

When you use Generators(see page 140) to build a structure, you are not placing specific tasks into your structure. Instead, you are creating a "skeleton" for your structure. Each time you open the structure, it is filled with the current content from Jira that fits the generator(s)[35] used to build the structure.

This means your structure will always reflect the most recent changes to Jira, and changes you make within your structure can also update Jira. It also means some restrictions need to be in place, so content isn't moved in a way that violates the Automation rules and makes it impossible to continue syncing content with Jira. Manual Adjustment makes it possible to bypass these rules, so you can customize any structure (regardless of how it's created) to fit your needs.

To learn more about how items can be moved within generated content, see Generator Scope(see page 198).

> Manual Adjustment works with structures built using Automation. It does not affect structures built manually or using Synchronizers.

3.4.11.5 Order of Operations with Manual Adjustments

The following Order of Operations is applied each time you open or refresh a structure with manual adjustments:

1. Run Generators – Structure runs your automation rules to import and organize Jira issues within your structure.
2. Manual Adjustments – Structure applies any manual adjustments over top of the generated content.
3. Transformations – Structure applies transformations after both generators and manual adjustments are made.

Manual Adjustments are applied AFTER generators. This means if anything changes within your Jira instance, it could also change (or even remove) your manual adjustment.

Here's an example:

1. You use a JQL Insert to add issues to your structure.
2. You then move Issue A under Issue X. You can't normally move issues within a JQL generator, so this requires a manual adjustment.
    - Since automation does not place issues into your structure, but rather creates rules for populating the structure (see Generators(see page 140) for more details), manual adjustments do not actually move issues. Instead Structure creates a new rule, explaining where those items should be in relation to the generated content. In this case, it creates a rule saying, "After all the generators are run, place Issue A under Issue X."
3. You make a change to Issue X, so it no longer fits within your Generator rules. The next time your generators run, Issue X will no longer be placed in the structure – so Issue A can't be moved beneath it. Issue A will remain in it's original location.

35 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4

Adding New Generators After Manual Adjustment

If you add additional generators to your structure after applying a manual adjustment, Structure will attempt to place manually adjusted items appropriately based on the existing and new generators.

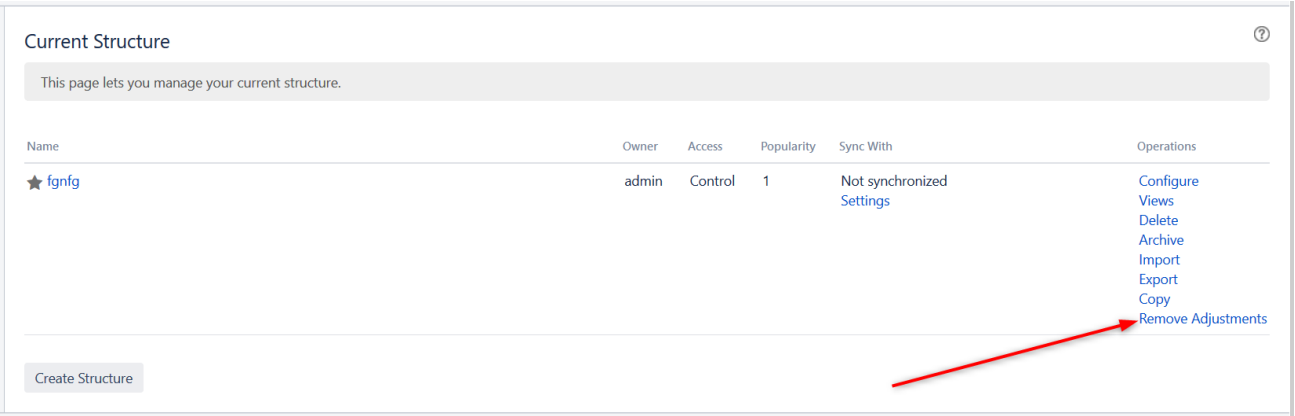
As you move it within the new set of generators:

- If it fulfills the generator's requirements, the manual adjustment will be removed and the item will be synced with Jira.
- If it does not fulfill the requirements, it will continue to be a manually adjusted item.

3.4.11.6 Undoing Manual Adjustments

To undo a single manual adjustment, simply drag the manually-adjusted item back to its original position.

To undo all manual adjustments, in the top menu go to **Structure | Manage Structure**, locate the structure you want to change, and click **Remove Adjustments**.



Removing all adjustments cannot be undone.

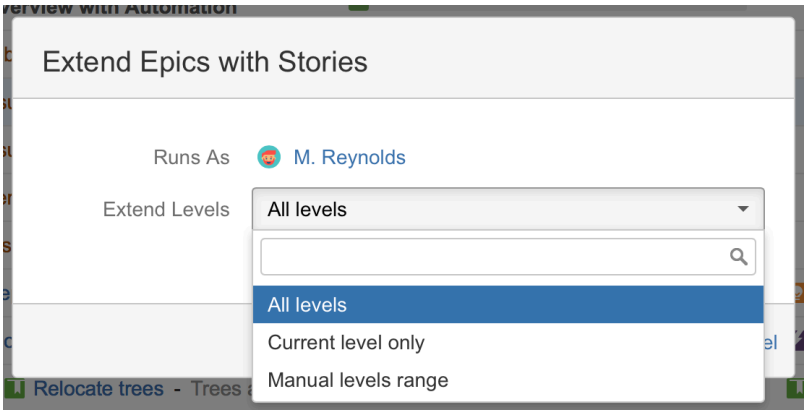
3.4.12 Managing Generators

3.4.12.1 Generator Scope

The scope of a generator is defined by its position in the structure and the **Levels** option.

- If you place the generator under the top-level root item (the structure's name), the generator will be applied to the whole structure.
- If you place it under some static item within the structure, the generator will only affect the descendants of this item.

To limit the scope further, you can set which levels the generator should be applied to within the generator's options dialogue – either when creating the generator, or by double-clicking the generator within the structure.



Most generators allow you to select from the following Levels:

- **All levels** - the generator will be applied to all descendants of the parent item.
- **Current level only** - the generator will only be applied on the level where the generator is added.
- **Manual levels range** - you can define the specific levels where the generator should work.

Manual Levels Range

The From and To fields define the range of levels to which the generator will be applied. The number entered into each field represents a level in the hierarchy, where 1 equals the level the generator is on, 2 is the next level down, etc.

Be Specific

If you wanted to pull in issues linked to the issues on level 2, set the **From** field to 2 and **To** field to 2. This will limit the generator to that specific row.

If the **To** field is set to 3, this would:

1. Pull in all issues linked to the issues in level 2, and
2. Pull in all issues linked to the new issues you just pulled in (because they will be placed on level 3).

You can also leave the From or To field blank:

- When the From field is blank, the generator is applied from the current level to the level indicated in the To field.
- When the To field is blank, the generator is applied to the level indicated in the From field and all levels below it.

> ⚠ For Group generators, the levels created by other Group generators are not taken into account when applying the specified manual levels limitation, unless the **Consider other groups** option is selected.

Building Hierarchies

The **Manual levels range** is especially useful when you want different levels in your hierarchy to have different types of relations. For example, you may want the top level and 2nd to be connected with issue links, the 2nd and 3rd with epic links, and the 4th level to be sub-tasks.

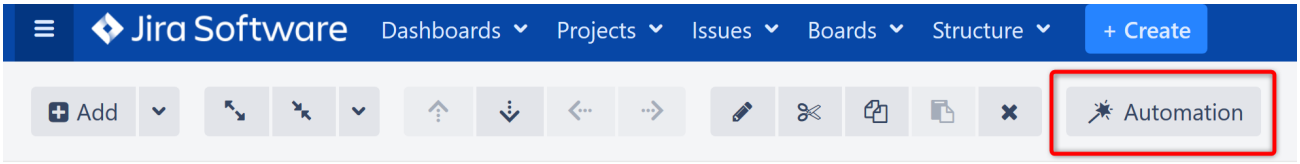
In this case, you will have three generators added under the root of the structure with the following **Levels** settings:

1. Linked Issues Extender (see page 149)working only on the top level - **Current level only**
2. Stories Under Epics Extender(see page 154) working on the second level - **Manual levels range**: from 2 to 2
3. Sub-tasks Extender(see page 152) working on the third level - **Manual levels range**: from 3 to 3

> If you built your entire structure using Automation, you will actually have four generators - the first generator will be an Insert generator.

3.4.12.2 Editing a Generator

To edit an existing generator, first switch on the Automation Editing mode by pressing the Automation button.



Next, locate the generator you want to edit within the structure.

Change Scope

To change the scope of the generator, simply move it to a new place in the structure, just as you would move any other item. You can use drag-and-drop, or copy/paste.

> You cannot move a generator under an item that was added by another generator.

Change Settings

To change a generator's settings:

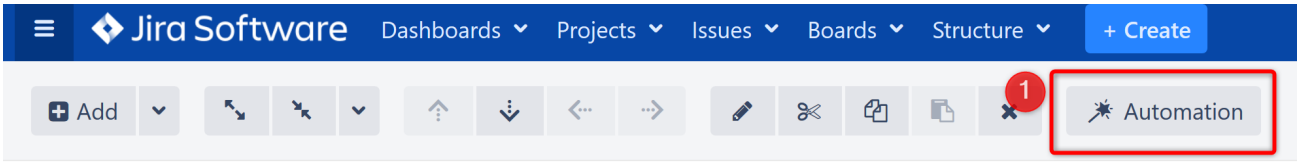
1. Double-click the generator or use the **Edit** button in the Structure toolbar.
2. Make the required changes and click **Apply** to save them.
3. Click the **Automation** button to hide generators (optional).

> To learn more about the options available for each type of generator, see the appropriate article in Types of Generators[36].

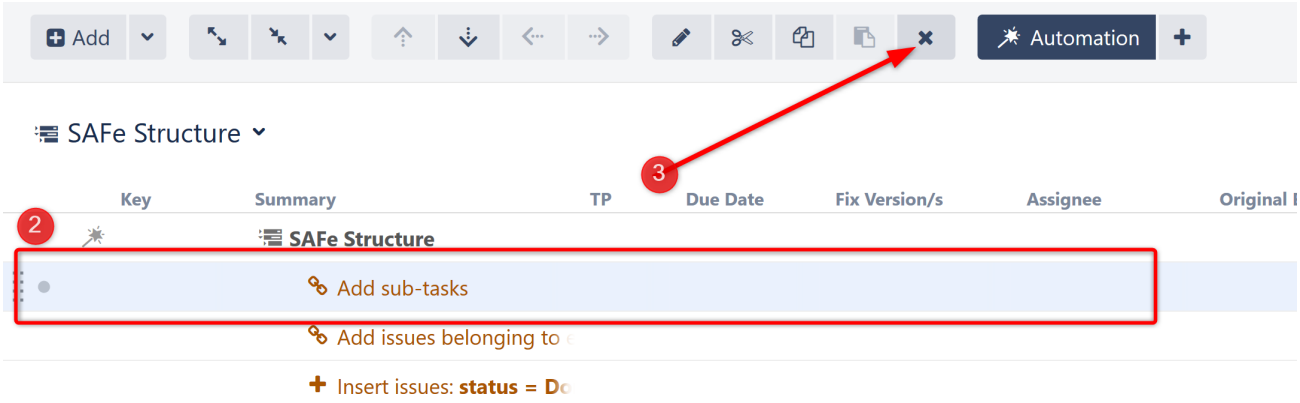
3.4.12.3 Deleting a Generator

To remove a generator from your structure, first switch on the Automation Editing mode by pressing the Automation button in the Structure Toolbar.

36 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4



Next, select the generator you want to delete and press **Delete**, or use the **Delete** button in the toolbar.

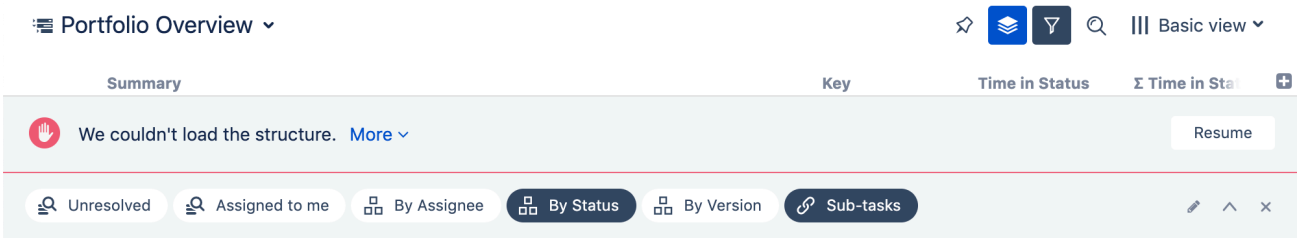


Click the **Automation** button to hide the generators (optional).

3.4.12.4 Automation Paused

To avoid unnecessary high resource consumption, Structure moderates the generation time for every structure by limiting it to a fixed value. If a structure is not generated within the existing time limit, the generation process is paused. This applies both to generators added as a part of the structure Automation settings and the additional transformations that maybe applied on top.

Whether the generation processed stopped because of the Automation or Transformations, a user will see the empty structure with only the skeleton visible and a notification banner will appear above the structure, letting users know generation is paused.



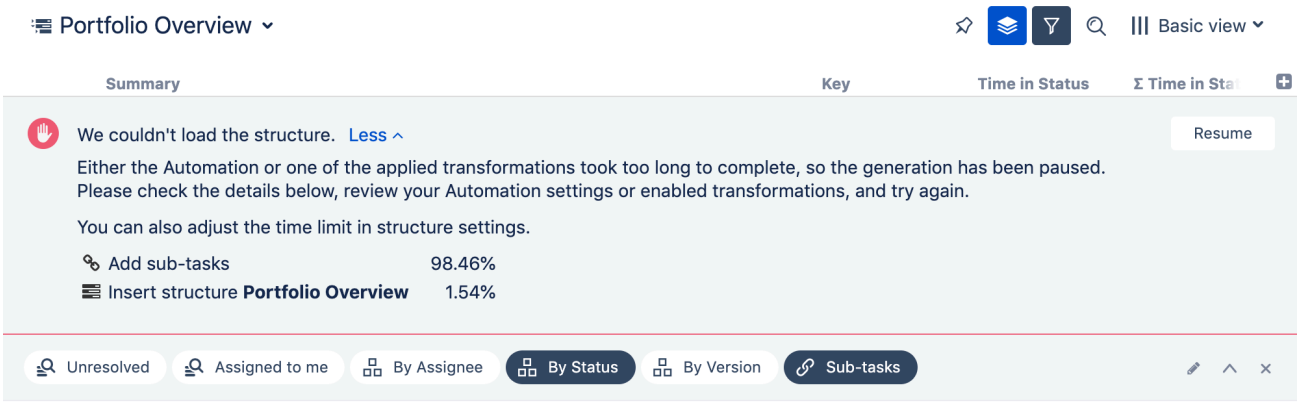
> If a timed-out structure is addressed in the 'structure()' JQL function, the JQL search of that query will return an error message.

Resuming Generation

To find out which generators or transformations are working too slow, click **More** on the notification banner. Additional information will be shown, displaying the overall percentage of time that each generator took before generation was paused. The highest number will indicate the slowest generator.

If the generation process stopped while running the generators, which are part of the Automation settings, you will see the list of all generators configured and the percentage of time they consumed. This structure will also be marked as paused (see sections below for details).

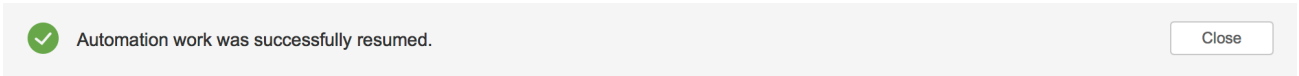
If the part of the structure created by Automation was loaded successfully and the timeout was reached while executing the transformations, you will see one Insert generator that represents the part of the structure that was added by Automation (Insert structure) and then you will see the list of all the transformations that finished successfully and the one on which the generation process was paused. This means if you had more transformations applied after this last one, they won't be mentioned in this list as they haven't been executed.



After reviewing and adjusting the automation and transformations, click **Resume** on the banner. If a structure still takes too long to generate, generation will be paused again, and the notification banner will update the 'More' section with relevant details. More actions from a user will be required, until all of the existing generators will work within the time limit.

> Deleting a generator or transformation from a structure doesn't resume generation. You must first click the 'Resume' button. This allows you to perform multiple actions before resuming.

When generation is resumed successfully, the structure will be updated with content and the banner's message will be changed accordingly. At this point, you can close the banner.

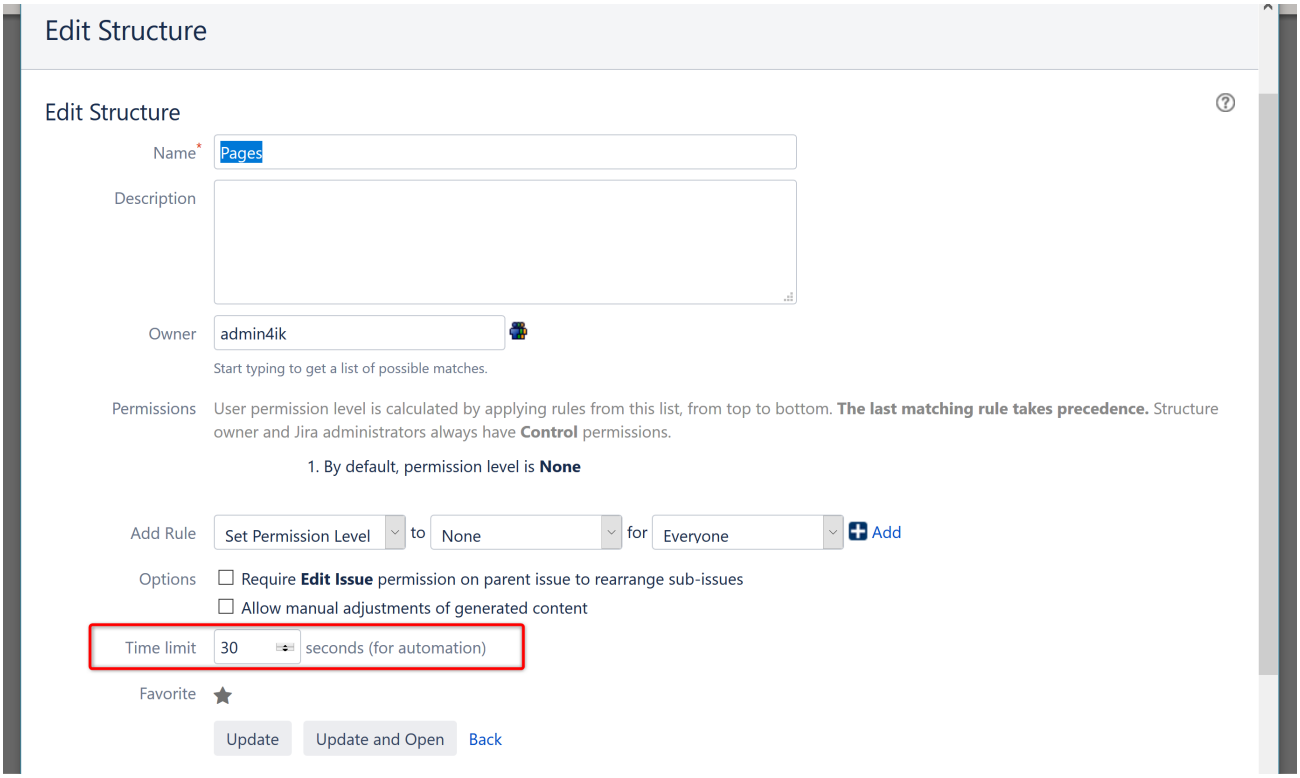


Changing the Allowed Generation Time for a Structure

If generation is paused by a generator or transformation that you consider reasonably configured, you can increase the default time limit of 30 seconds and let the structure generate for a longer time period.

To edit the time limit:

1. Go to **Structure** | **Manage Structures** in the Jira menu. Locate the structure you want to adjust (in most cases, this will be the Current Structure).
2. Click **Configure.**
3. Adjust the **Time limit** to the number of seconds you want generation to wait before timing out.
4. Click **Update** to apply the new settings.



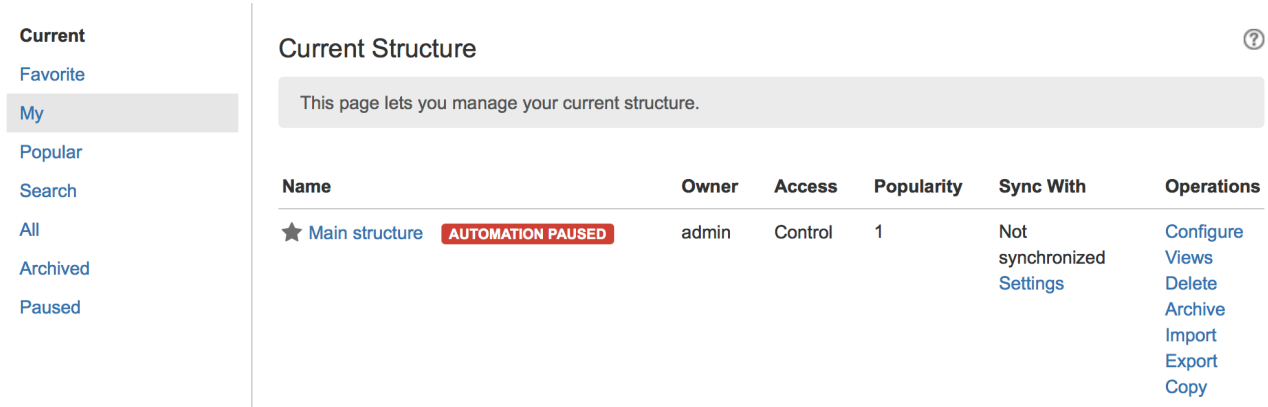
Time Limit Guidelines

When changing the **Time limit**, keep the following in mind:

- A generation time limit can't be less than 5 seconds or more than the system-wide hard limit
- Although the value is entered in seconds, the limit can be set to several minutes
- 'Control' structure permission is required to change the time limit

Identifying a Paused Structure

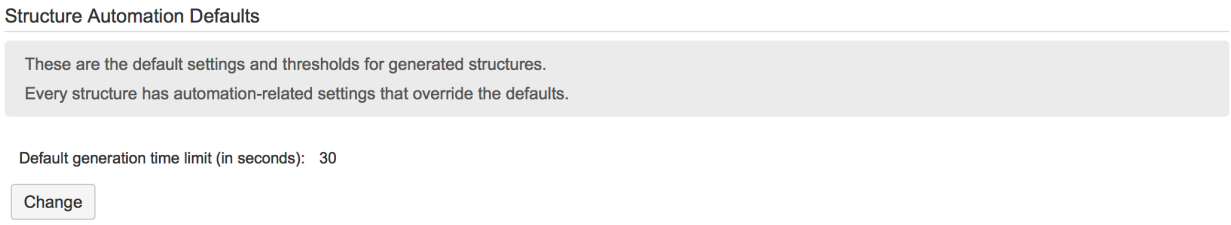
If generation is paused in a structure, the **AUTOMATION PAUSED** indicator will appear next to the structure's name on the Manage Structures screen.



You can quickly locate paused structures by clicking the **Paused** option in the left menu.

Changing the Default Generation Time Limit

To change the default generation time limit for all structures, go to **Administration** | **Structure** | **Defaults**. Look for the the **Structure Automation Defaults** section and click **Change**.



Only Jira administrators can change this setting.

If the time limit was manually set for a specific structure, it will not be changed to the default one. Only structures using the default time limit will be affected.

System-wide Hard Limit

The system-wide generation time limit is initially set for 10 minutes. It can be adjusted by using Script Runner or other similar tools, or by changing the `structure.gfs.generationTimeHardLimit` property, with the Structure Dark Feature and Fine Tuning Interface(see page 974). The Generation Time Hard Limit setting accepts an integer number in seconds.

The generation time limit in a structure is only taken into account if it is less than the system-wide hard limit; otherwise it is ignored and the system-wide hard limit is used instead.

3.4.12.5 Order of Operation for Generators

Generator Types

When multiple generators are present in a structure, they are run in the following order, based on their generator type:

1. Insert generators
2. Extend generators
3. Filter generators
4. Group generators
5. Sort generators

Generators of the Same Type

If there are multiple generators of the same type:

- Group generators - grouping is applied from top to bottom, starting with the top-most Group generator

- Sort generators - sorting is applied from bottom to top, starting with the bottom-most Sort generator
- This does not affect the results of Insert, Extend or Filter generators

To change the order generators of the same type are run, simply move them up or down in the structure.

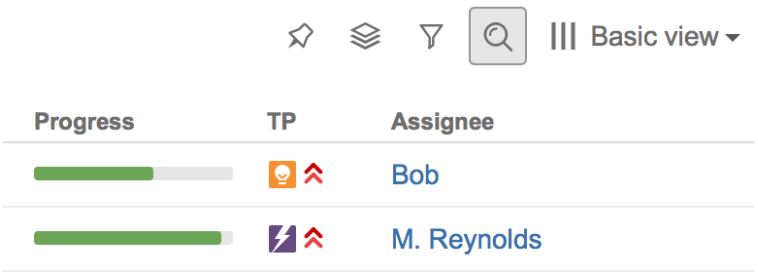
> ⚠ Moving generators up or down in the structure does not affect which types of generators are run first.

3.5 Search

The Search feature allows you to:

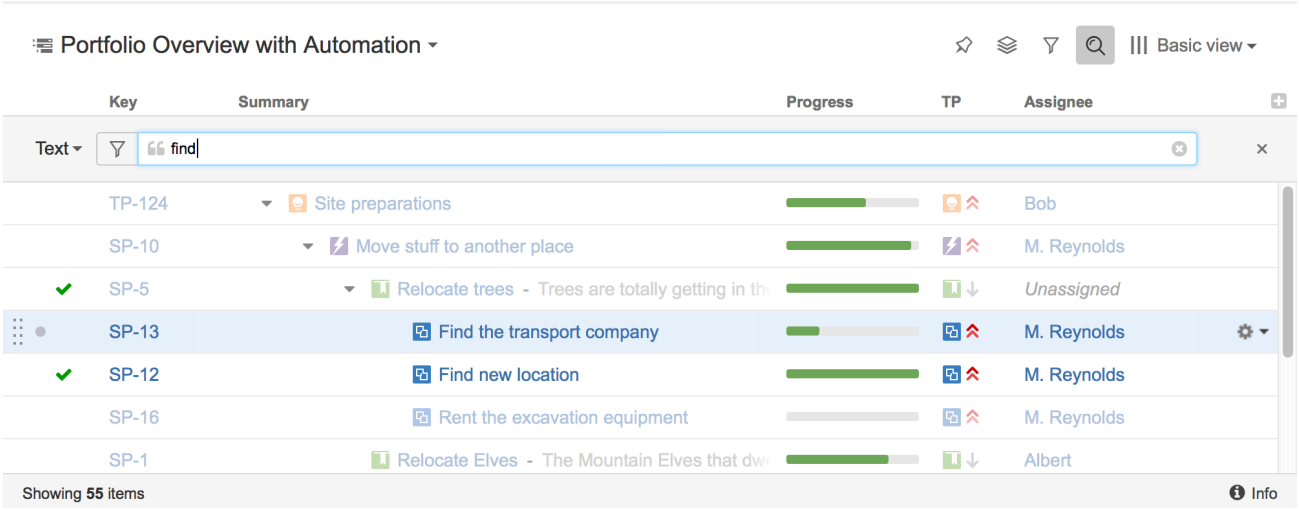
- Find and highlight issues in your structure
- Filter(see page 207) your structure so it only displays specific issues

To access Search function, click the **Search** button on the Structure Panel Toolbar.



The Search panel will appear below the toolbar. As you enter a query into the search field, results are filtered immediately, and then refined as you keep typing.

All non-matching items are grayed-out, in order to highlight your search results.



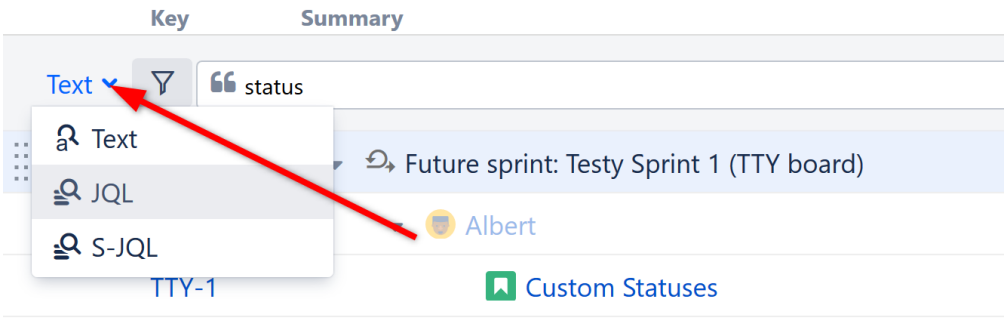
If data changes on the server, search results are automatically refreshed for the structure. So issues can be hidden and shown in the structure in real time.

Keyboard Shortcuts

- Move between matching items**: Ctrl+Alt+]** and **Ctrl+Alt+[**
- Turn on Search (or switch search mode): **Ctrl+Alt+/**
- Cancel Search & close Search panel: **Escape**

3.5.1 Search Modes

You can search for issues within the current structure using a Text(see page 206), JQL(see page 207), or S-JQL(see page 207) query. To switch between these modes, click the name of the currently-selected mode and select a new mode from the menu.



3.5.1.1 Text Search

Text search is selected by default. In this mode, you can specify the following search conditions:

Condition Type	Example	How it works
Simple text	*structural hierarchy*	Looks for items that have **all** mentioned words in the **Summary** field. Each word must be present in the summary or name, or the summary must have a word that **begins with** the specified word. The words may appear in any order.
Quoted excerpt	*"the quick brown fox"*	When quotes are used, the search looks for the entire phrase in the summary or name fields.
Issue keys	*MARS-1, MARS-331*	If the text looks like one or more issue keys (delimited by comma or white space), the search will return exactly these issues (if they are in the structure).

Structure relies on the Jira search engine to run text searches. The engine is based on Lucene index, which has a few peculiarities that may cause unexpected results. For example, short words may not be found. The result also depends on the Indexing Language specified in the Jira General Configuration.

3.5.1.2 JQL Search

JQL (Jira Query Language) lets you specify arbitrarily complex conditions to find very specific issues.

When entering a JQL query, auto-complete will suggest fields, operators and values as you type. When you have a valid JQL query in the search field, a green checkmark icon will appear beside your search. When the JQL is invalid or not complete, the red exclamation icon is displayed.

To learn more about JQL, see the Jira documentation[37].

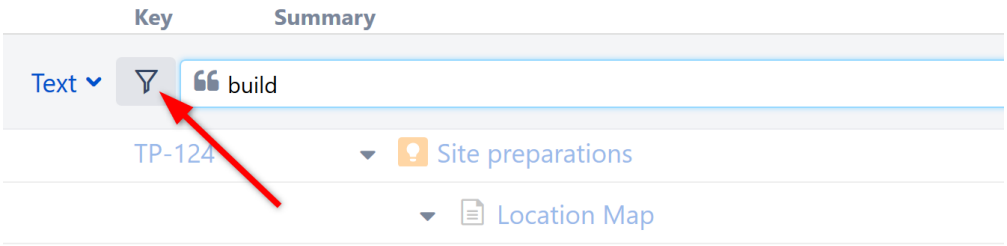
3.5.1.3 S-JQL Search

S-JQL (Structured JQL(see page 434)) is a special language that allows you to search for issues by their relationships in the current structure. For examples, `root` matches all top-level issues, `root or child of root` matches the first two levels, and `child of [priority = Critical]` matches all children of critical issues. To learn more about S-JQL conventions, see Structured JQL(see page 428).

As with a JQL search, the checkmark or exclamation indicators will let you know whether the query is valid or not.

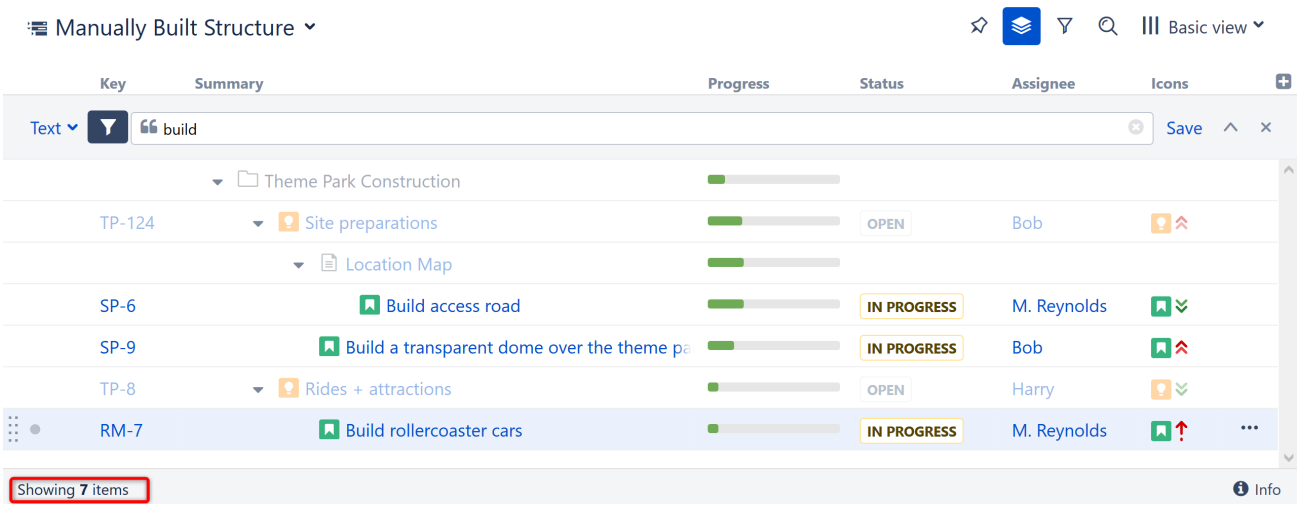
3.6 Filter

When using Search, items that do not match the search criteria are grayed out, but remain in your structure. To remove those items (so you will only see items that match your query), click the **Filter** button to the left of the search field.



Once Filtering is turned on, only those items that match your query and their parent items will be visible in the structure. Parent items are kept to preserve the hierarchy view, but they are grayed out.

37 http://confluence.atlassian.com/display/JIRA/Advanced+Searching

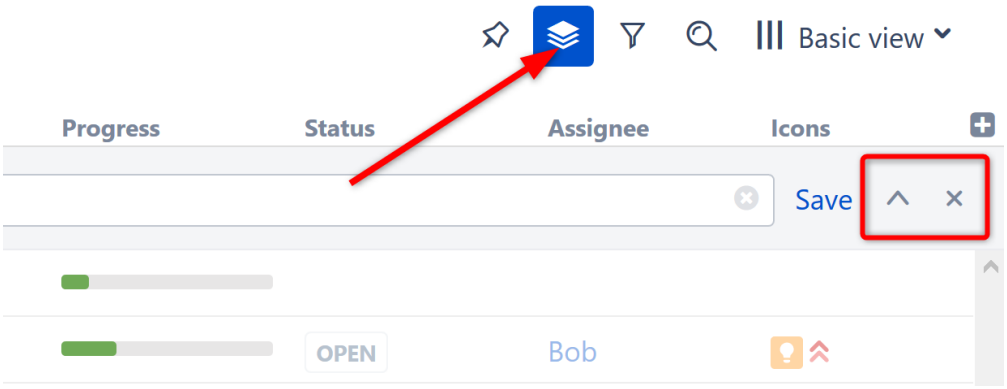


In the status bar at the bottom, you will see an updated items count.

You can apply the filter to any text, JQL or S-JQL search(see page 205).

3.6.1 Filter Is a Transformation

When you apply the filter, you are also creating a Transformation(see page 216). The Transformations button is highlighted, showing that a transformation has been applied.



3.6.1.1 Remove/Hide a Filter

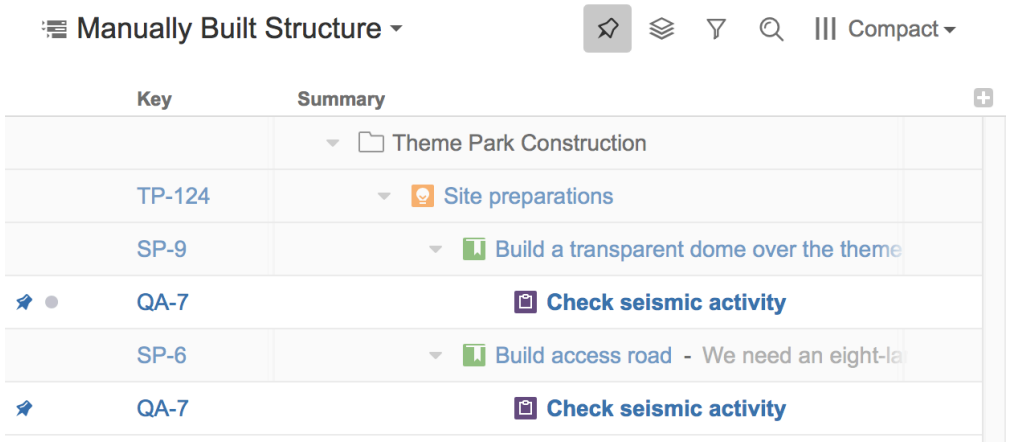
To remove the filter, click the close button (**x**) on the right side of the search field.

To hide the filter bar while keeping the filter applied, click the arrow button next to it. Once you do this, you will need to click the Transformations button again to remove the filter.

Filtering mode remains active even if you navigate to another page.

3.6.2 Pinned Item Mode

To focus on a specific item and only view parts of the structure that relate to that item, click the **Pin** icon on the panel toolbar. If an item appears more than once within the structure, every instance of the item will be put in focus.



Structure Panel on the Issue Page(see page 90) automatically pins the issue being displayed, so you only see the relevant part(s) of the structure.

3.6.2.1 What is Displayed in Pinned Item Mode?

When a structure is in Pinned Item Mode, only the following items are displayed:

- The pinned item itself (all instances of the item within the structure)
- All parent items of the pinned item, up to the top-level item
- All sub-items of the pinned item, down to the deepest level

Items that are "siblings" or located somewhere else in the hierarchy are not displayed.

3.6.2.2 Turning Pinned Item Mode On and Off

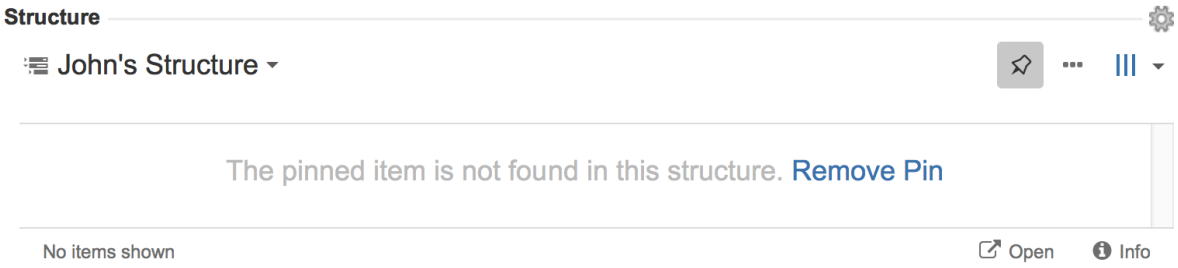
To turn Pinned Item Mode on or off, click the Pin button on the panel toolbar or press **Ctrl+.** on your keyboard.

When Pinned Item Mode is turned on:

- On Structure Board - whatever item is selected in the structure will be pinned. You can pin any issue on the Structure Board.
- On a Project Page or Agile Board - whatever issue is selected in the project/board's issue list will be pinned. To pin a new item, select a different item in the issue list.
- On an Issue Page - the corresponding issue will always be pinned. It it not possible to pin any other item when viewing a structure on an issue page.

Pinned Item Not Found

If the pinned item is not in the selected structure, you will receive the following message:



If you see this message you can:

- Click **Remove Pin** or click the **Pin** button to view the entire structure. If you want to add the selected issue to the structure, click the **Paste** button in the toolbar.
- Click the structure's name and select a new structure.
- If you are viewing the structure from a Project Page or Agile Board, select a new issue from the issue list.

3.6.2.3 Limitations Imposed by Pinned Item Mode

When an item is pinned, there are some restrictions to how items can be manipulated within the structure:

- Items above the pinned item (its parent/ancestors) cannot be moved or deleted
- You cannot add items above the pinned item or as siblings to the pinned item
- Items beneath the pinned item can be moved or deleted, and new items can be added below the pinned item

> Even though you can't move parent items while in Pinned Item Mode, you still can select them, edit or apply Jira operations.

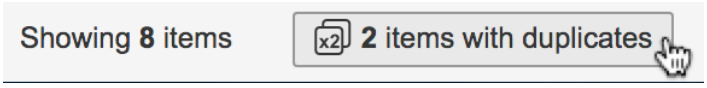
3.6.3 Identifying Duplicate Items

Structure allows you to have multiple instances of the same item in a single structure, so you can quickly visualize when a single issue impacts several others, or a single bug affects multiple issues.

Structure also makes it easy to identify and manage duplicates within your structure.

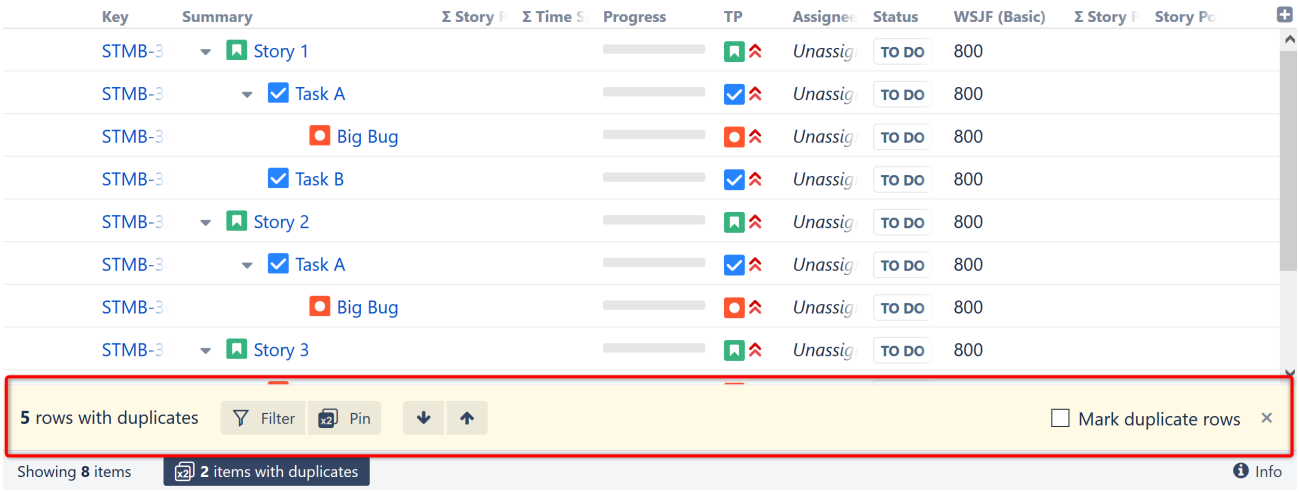
3.6.3.1 Finding Duplicates

If any items appear more than once in a structure, a duplicates counter will appear in the status bar.



> The duplicates counter works for the visible part of a structure only. If you apply filters on top of the existing structure (for example, transformations), and these filters hide all but one of a duplicated item, that item will no longer be counted.

Clicking the duplicates counter will open the Duplicates panel, which allows you to highlight, filter, pin or quickly navigate between duplicate items.



Duplicate Counts

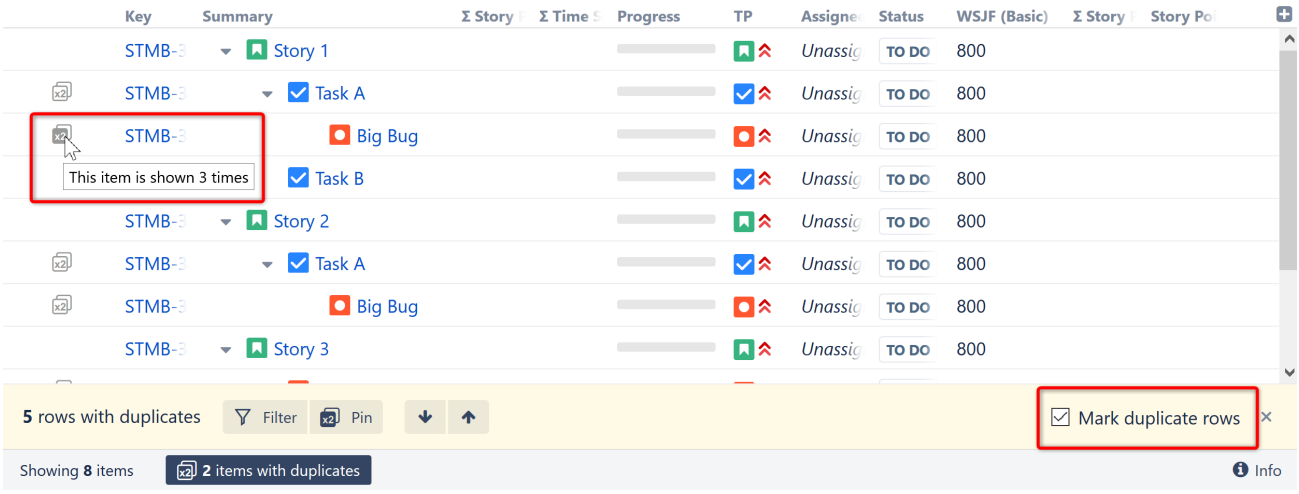
The duplicates counter shows the number of items that have duplicates. In the example above, there are two duplicating items (Task A and Big Bug).

The Duplicates panel shows the number of rows those items produce. In the example above, there are five rows of duplicates:

- Task A appears twice
- Big Bug appears three times

3.6.3.2 Mark Duplicate Rows

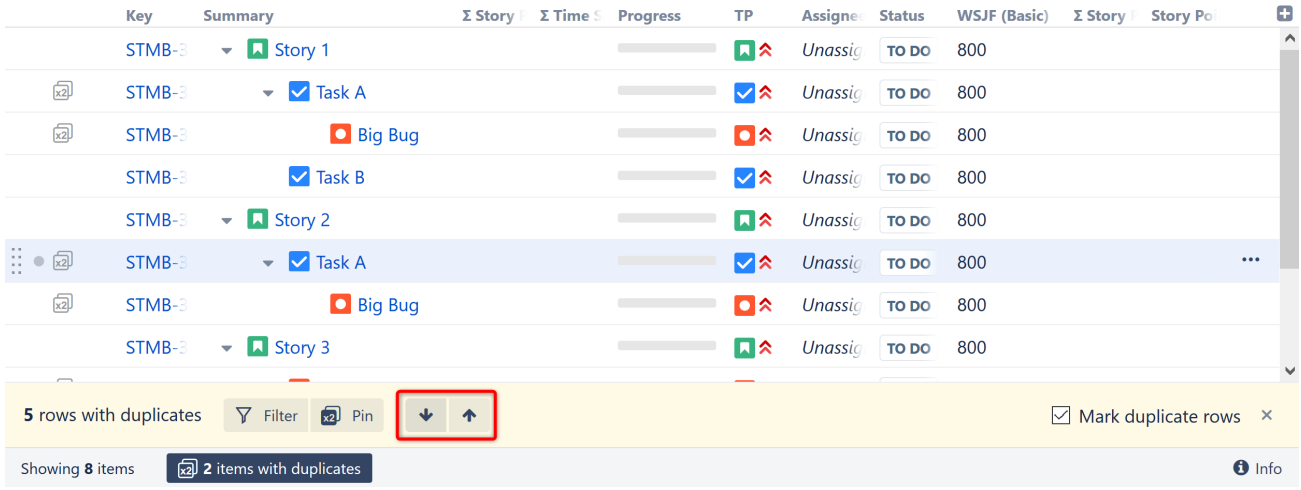
To highlight duplicate items, select the 'Mark duplicate rows' option. This places a duplicate icon (with ‘x2’ written on it) to the right of each duplicate row. If you hover over that icon, you'll see how many instances of this item are shown in the structure.



Once you highlight duplicate items, the highlighting stays even if you close the Duplicates Panel. To remove the highlighting, open the panel and clear the 'Mark duplicate rows' checkbox.

Navigating Between Duplicates

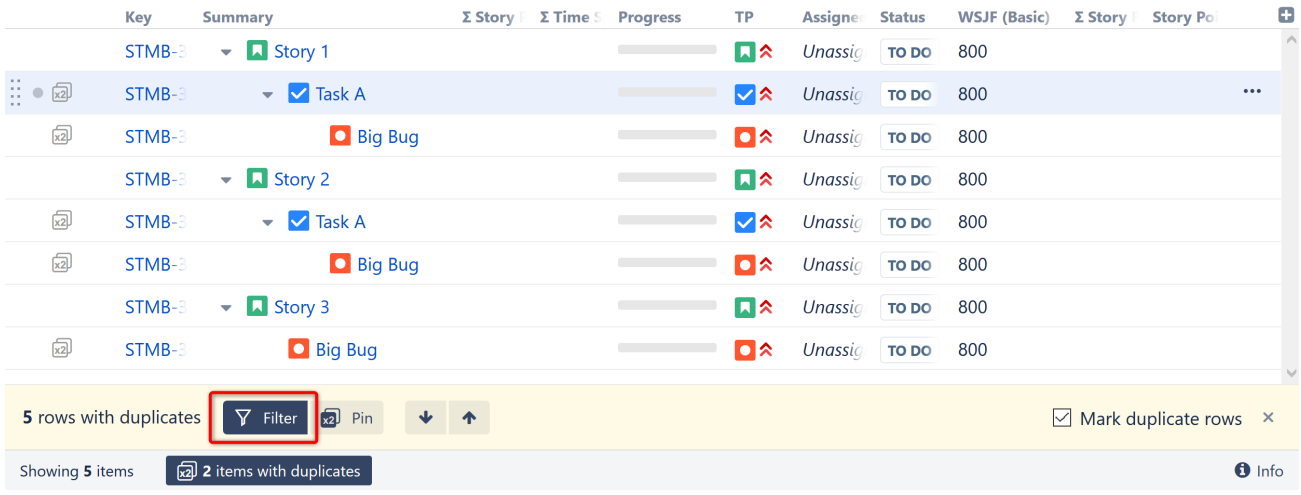
Clicking the “Up” and “Down” arrow buttons in the Duplicates panel moves the focus to the next/previous duplicate row.



When you navigate to a duplicate row that is in a collapsed part of a structure, this part of the structure will be expanded so the duplicate is visible.

3.6.3.3 Filtering by Duplicates

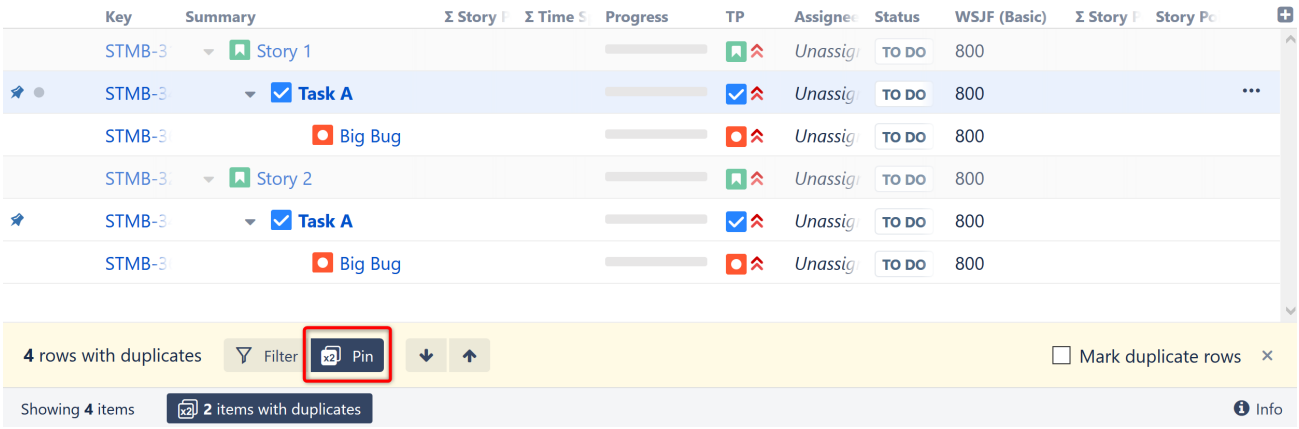
Clicking the Filter button hides everything in the structure except duplicate items and their parents.



> When the Duplicates panel is closed, the Duplicates filter is disabled automatically.

3.6.3.4 Pinning Duplicates

The Pin button in the Duplicates panel hides everything except the selected item, its parents and its children. If the pinned item is a duplicate, all instances of that item and its parents/children will be displayed.



You can quickly pin items by simply clicking the 'x2' icon beside any duplicate item.

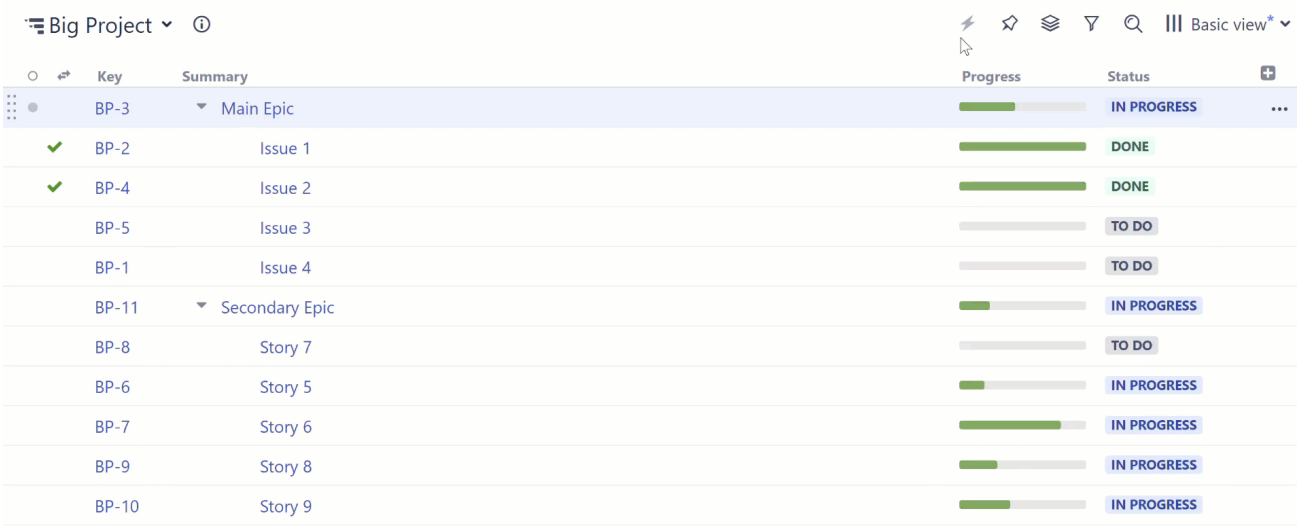
3.6.4 Filter by Field Transformation

The Filter by Field transformation allows you to quickly filter the issues in your structure based on values in a Jira field. This allows you to focus on specific issues, such as issues from the next fix version, issues assigned to specific users, issues in a specific set of statuses, and more.

3.6.4.1 Applying a Filter by Field Transformation

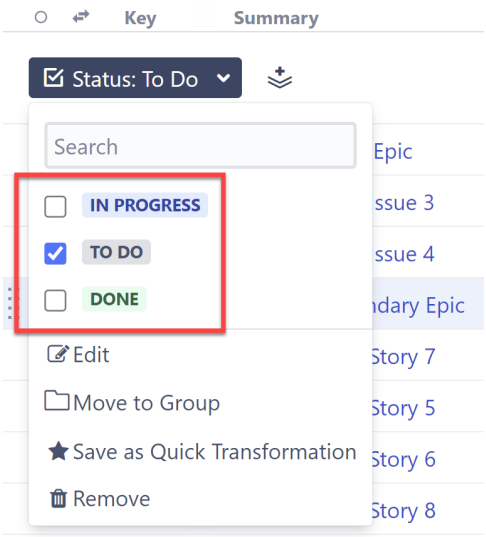
To apply a Filter by Field transformation:

1. Click the Transformations icon.
2. Click **Add Transformation**.
3. Select **Filter > Field...**
4. Select the issue field you want to filter by.
5. Select the field values to be used for the filter - any issues with the field values selected here will remain in the structure.



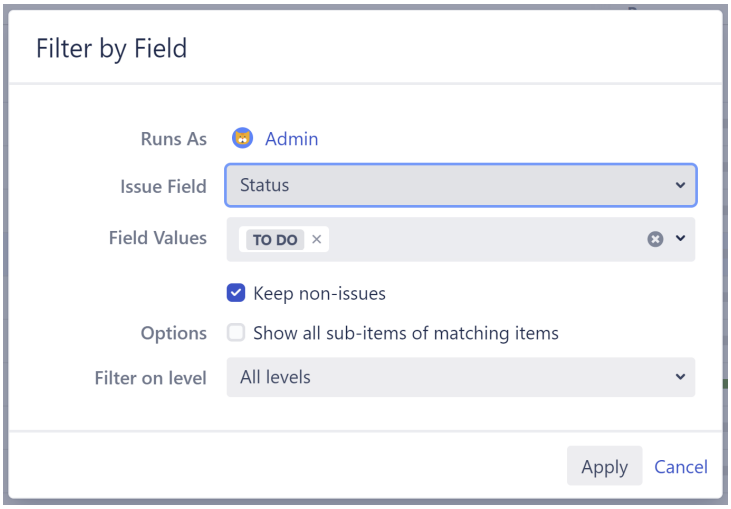
3.6.4.2 Change Filter Values

To change the selected values for an existing filter, click the down arrow next to the filter's name and make any necessary changes to the selected fields.



3.6.4.3 Additional Options

To apply additional options to a Filter by Field transformation, click the down arrow next to the filter's name and select **Edit**.



On the options screen, you can change the field, select different values, or apply any of the following options:

- **Keep non-issues** - When checked, non-issue items, such as folders, will remain in your structure regardless of whether or not they match your filter criteria.
- **Show all sub-items of matching items** - If this option is selected, all issues that match your filter criteria will be included in the structure, along with any sub-items of those issues.
- **Filter on level** - Allows you to apply the filter to specific levels within the hierarchy.

3.6.4.4 Filter by Field Transformation vs Generator

Filter by Field is available as a Transformation and a Generator.

- The Filter by Field transformation allows you to temporarily filter items in a structure to focus on very specific issues. The filter is applied locally, so anyone else viewing the structure is unaffected. This is useful when you need to focus in on specific issues without changing the underlying structure for yourself or anyone else.
- The Filter by Field generator(see page 167) is a more permanent solution, which should only be used when you always want to see just the filtered results in your structure. This filter will be applied every time the structure is opened, and when the structure is shared.

3.7 Transformations

Transformations allow you to reorganize the issues in your structure, in order to focus on specific types of issues or issue properties. For example, transformations can be used to Filter(see page 207) out all but those issues assigned to the current user or to sort issues by progress (or both!).

Sorry, the widget is not supported in this export.
But you can reach it using the following URL:

https://www.youtube.com/watch?v=IW0JTizto58

3.7.1 Transformations vs. Generators

Transformations use the same types of functions as generators[38]. However, there are a few key differences:

- Transformations make local adjustment to the structure, without changing it for everyone else. If someone opens the same structure while you have a transformation applied, they will still see the original structure.
- Transformations can only be applied to the whole structure, while generators can be inserted under a folder or manually-added issue.
- There is no "Insert" transformation, because transformations are applied to the issues already in a structure.

3.7.1.1 Which should you use?

- If you want to reorganize the issues you see, without affecting anyone else's view, use Transformations.
- If you want to apply a temporary change, transformations are also preferable, because they are easily switched on or off and can be saved for quick access.
- If you are organizing issues in a way that others would benefit from, consider using generators and saving it as a new, shared structure.

3.7.2 Available Transformations

The following types of transformations are available in Structure:

- Filter
- Sort
- Group
- Extend

For more details on how each of these works, please see the documentation on Types of Generators[39].

> ⚠ The Filter by Sprint transformation only affects folders, not issues, and it is applied to the whole structure. The Filter by Sprint generator is only applied to embedded sub-structures.

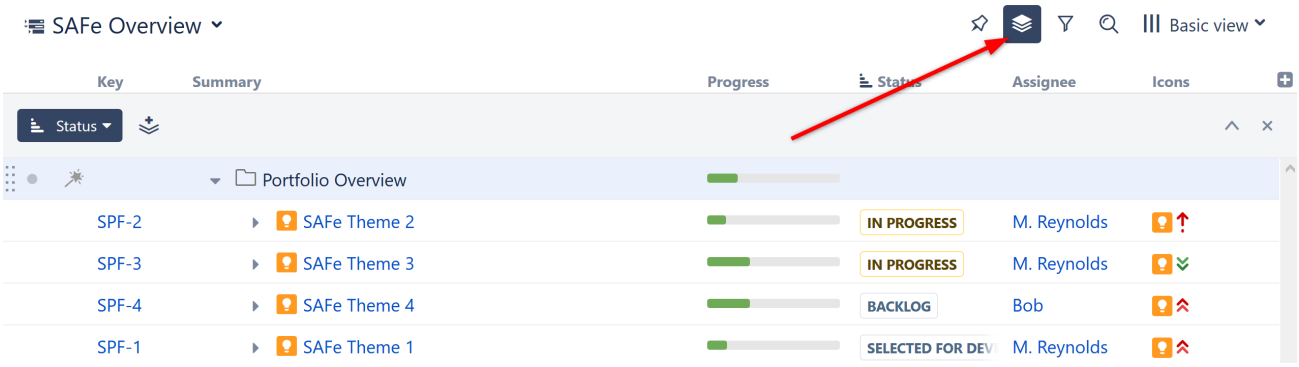
3.7.3 Working with Transformations

- Using Transformations(see page 216)
- Quick Transformations(see page 220)
- Default Quick Transformations(see page 228)

3.7.4 Using Transformations

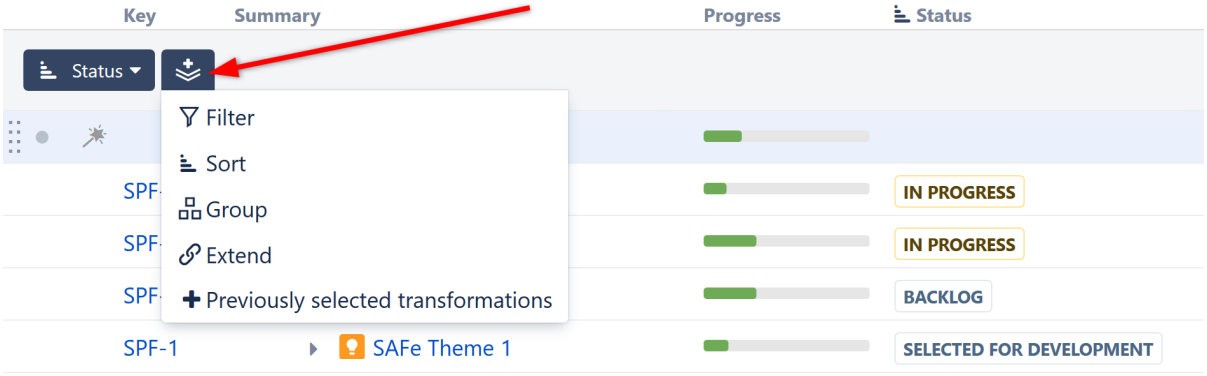
All transformations can be added and modified on the Transformations Panel. To access it, click the **Transformations** button in the panel toolbar. If any transformations are currently applied, they will appear in the panel.

38 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4
39 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4



3.7.4.1 Create a New Transformation

To add a new transformation, click the Add Transformation button and select one of the available transformations.



Transformations function in much the same way as their corresponding generators. For more information about how each works, see the documentation on Types of Generators[40].

> If you add transformations and then switch structures using the Structure selector, you can apply the same transformations to the new structure by selecting **+Previously selected transformations**.

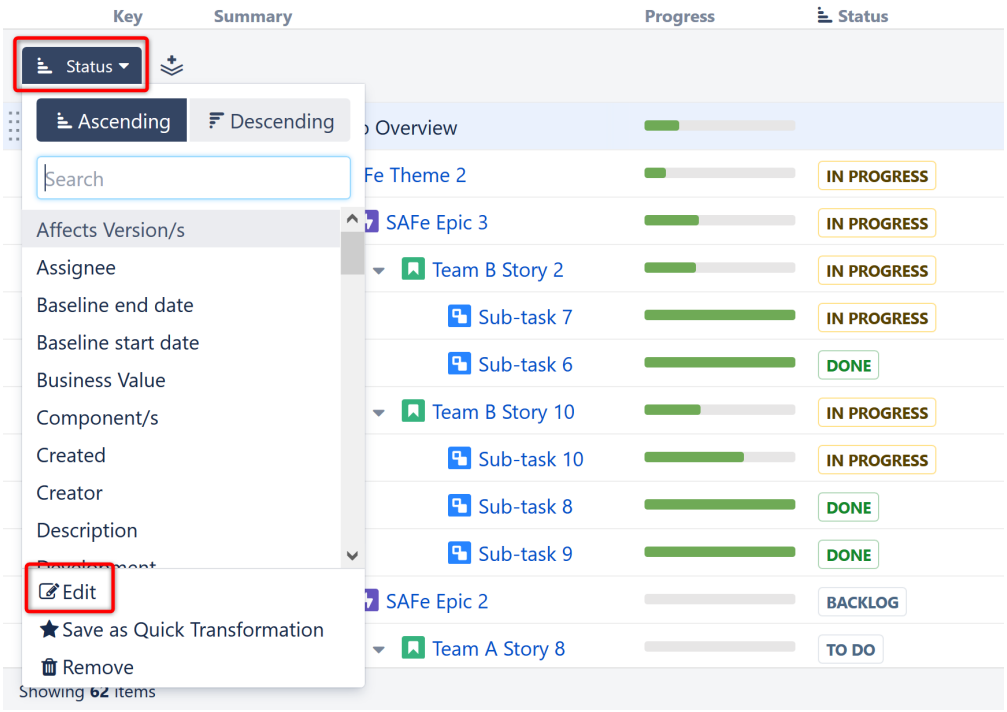
3.7.4.2 Manage Transformations

Edit Existing Transformations

To edit an existing transformation, click the transformation name in the Transformations Panel.

You can select a new attribute within the drop-down menu, change the order (for sort), or select **Edit** at the bottom of the drop-down to see advanced options.

40 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4



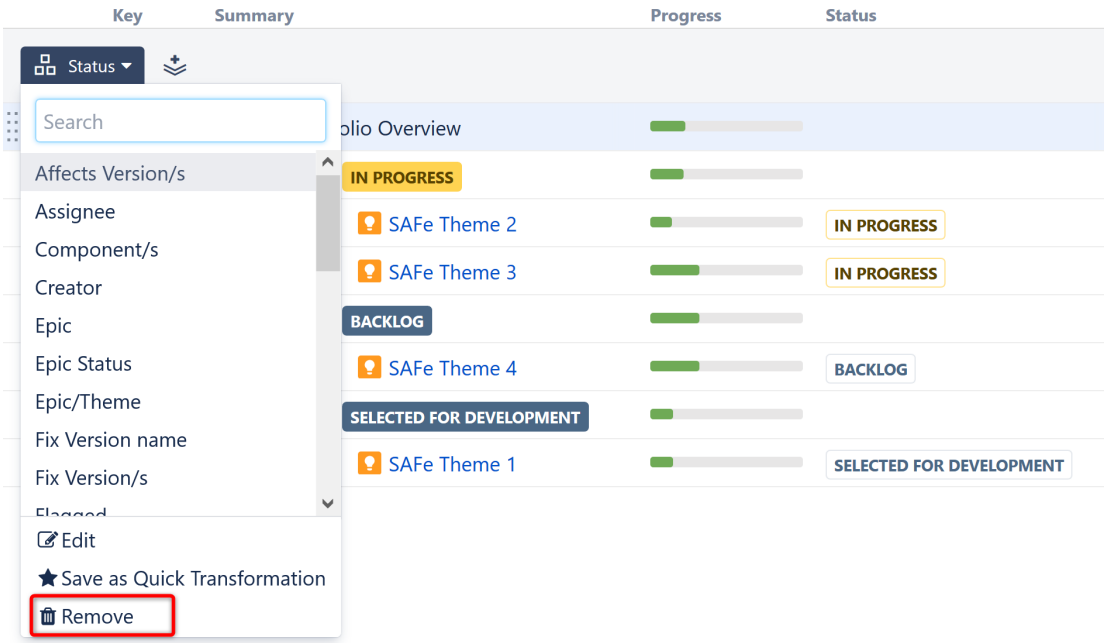
For more information about the advanced options available for each transformation, see the documentation for the corresponding generator[41].

You can quickly change a Sort transformation by clicking another column header – the structure will be sorted by the new column, in ascending order. See Sorting and Filtering(see page 220) to learn more.

Remove a Transformation

To remove a transformation, click the transformation name and select **Remove** from the bottom of the drop-down menu.

41 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4



Hide the Transformations Panel

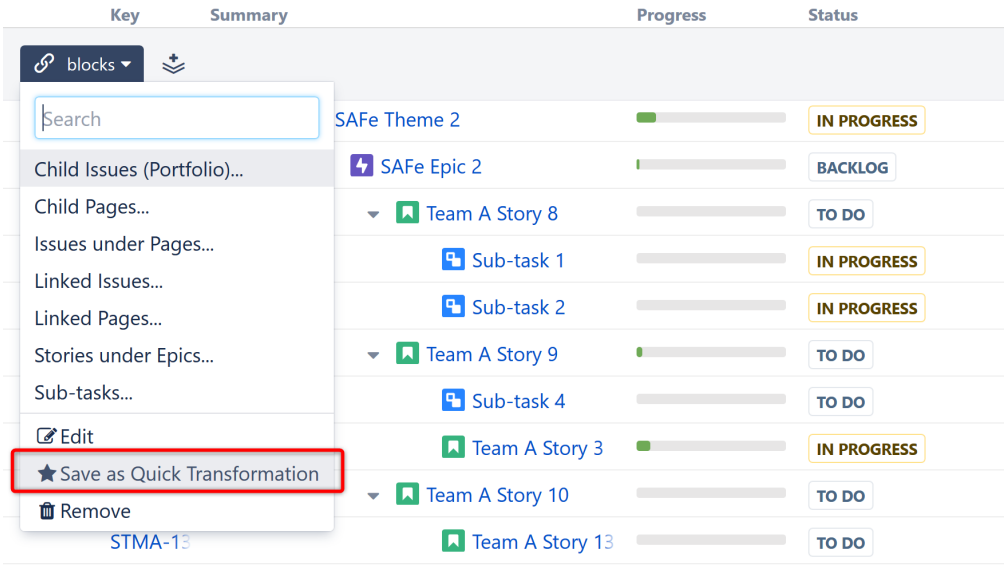
After you've configured your transformations, you can hide the panel without removing the transformations. Click the **up arrow button** on the right side of the Transformations Panel to hide it.

Remove All Transformations

To remove all transformations, click the close button (**x**) on the right side of the Transformations Panel.

Save Transformations

If you frequently use the same transformations, you can save them as Quick Transformations(see page 220). Click the transformation you want to save, and select **Save as Quick Transformation** at the bottom of the drop-down menu.



You must have Control permission for a structure to save Quick Transformations.

3.7.4.3 Sorting and Filtering

Sort and Filter transformations can be applied very quickly, without opening the Transformations Panel.

To **sort** your items, simply click the header of the column you want to sort by. Your structure will be sorted in ascending order, on every level.

- To sort in descending order, click the column header again.
- To change the scope, choose the Edit option(see page 217).
- To remove the sorting, click the **Summary** column header.

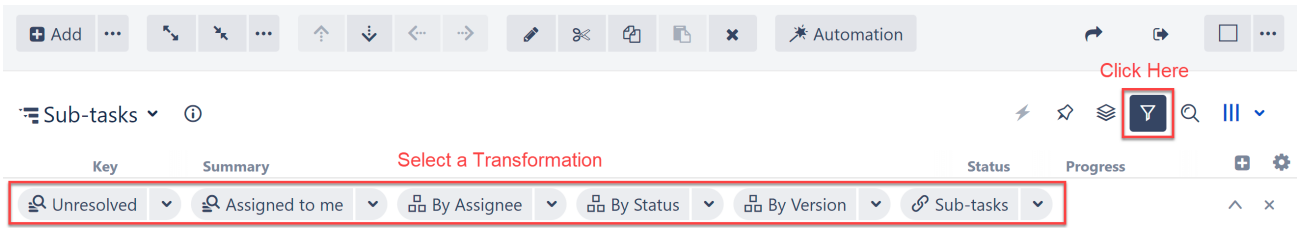
To apply a **filter**, you can run a search(see page 205) and then filter(see page 207) out non-matching items.

3.7.5 Quick Transformations

Quick Transformations allow you to apply commonly used transformations with the click of a button.

3.7.5.1 Activate a Quick Transformation

To apply a quick transformation, click the Quick Transformations button to open the Quick Transformations panel, and then select the transformation you want to apply. That's it!



You can add as many transformations as you need. Structure will remember the selected transformations, so the next time you open that structure, the transformations will already be applied.

> The Quick Transformation panel shows transformations that are associated with the current structure. See Defining Quick Transformations(see page 220) to learn how to add your own custom transformations.

Default Transformations

If the list of quick transformations was not customized for the displayed structure, the default quick transformations will be shown. Default transformations are also shown in the Secondary panel when displaying query results or other non-structure content.

To learn more, see Default Quick Transformations(see page 228).

Order of Operations

Quick transformations are applied in the order you select them.

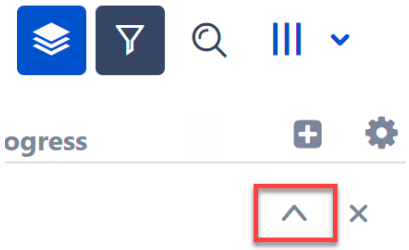
For example, if you click **By Assignee**, **By Status**, and then **By Version**, it will:

1. Group the first level by Assignee
2. Group the second level by Status
3. Group the third level by Version

To change this order, deselect the transformations and select them again in a different order.

Hide the Quick Transformations Panel

After you've applied quick transformations, you can hide the panel without removing the transformations. Click the **up arrow button** on the right side of the Quick Transformations panel to hide it.

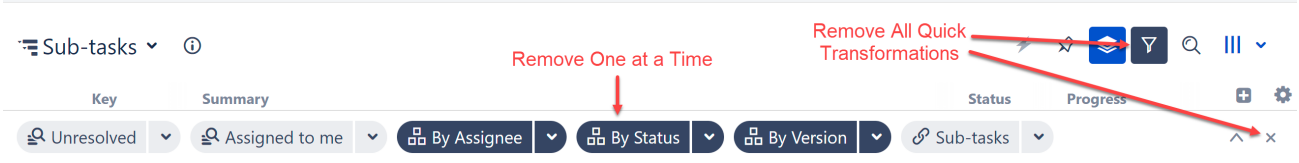


When the panel is hidden but quick transformations are applied, the Quick Transformations button will become blue .

3.7.5.2 Deactivate a Quick Transformation

To remove a transformation, deselect the transformation in the Quick Transformations panel.

To remove all quick transformations, click the "x" button on the right side of the Quick Transformations panel or click the Quick Transformations button.



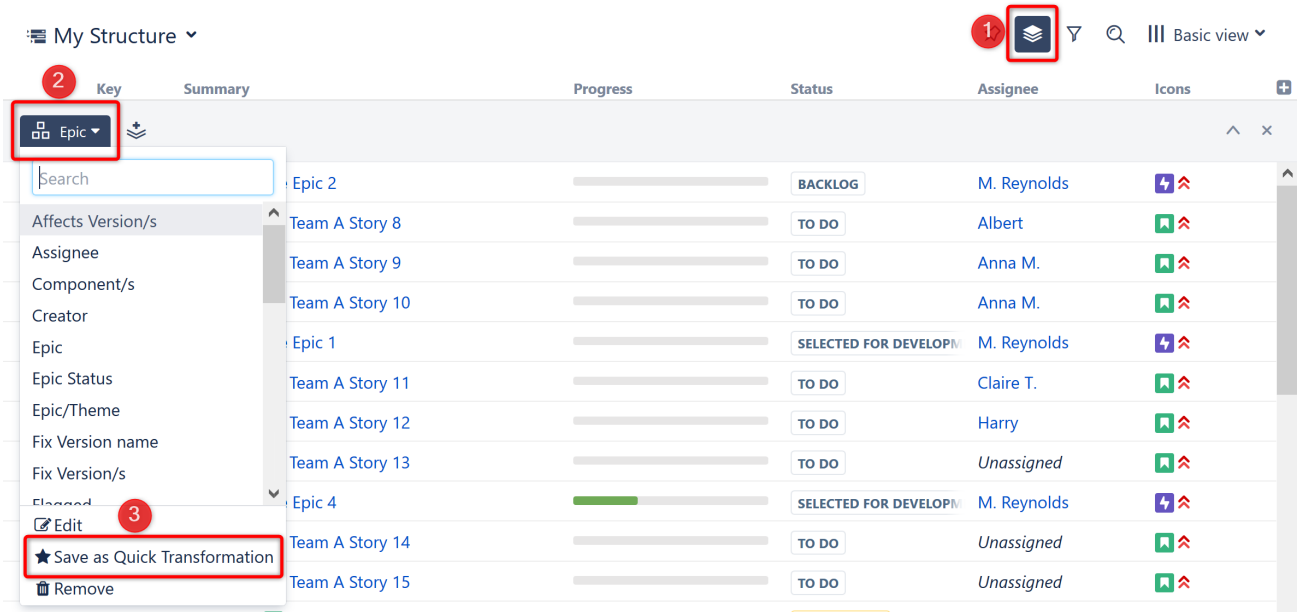
You can use keyboard shortcuts to toggle quick transformations, based on their position in the transformations list. The shortcut is **Q** and then the number (**1—9**), typed in quick succession.

3.7.5.3 Defining Quick Transformations

Quick transformations can be customized for each structure by anyone who has **Control** access to the structure.

To create a quick transformation:

1. Open the Transformations panel(see page 216)
2. Create your transformation
3. Click **Save as Quick Transformation**

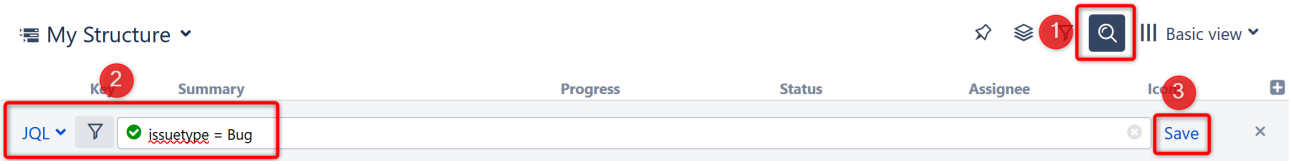


If you don't see a way to add a quick transformation, you likely do not have the appropriate permissions for that structure. You should ask the structure's owner to create the quick transformation.

Adding a Quick Filter

Filter quick transformations can also be created from a search:

1. Open Search(see page 205)
2. Enter your query (using Text, JQL, or S-JQL search)
3. Click **Save**



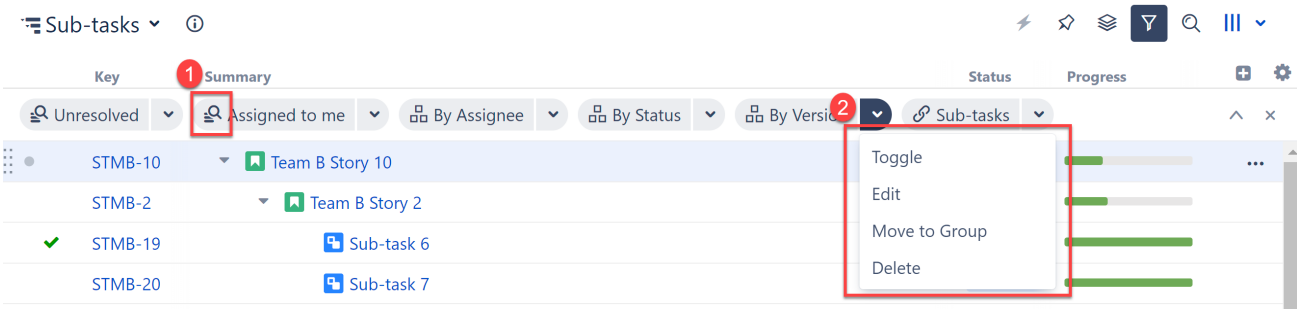
After clicking **Save**, you will be given the opportunity to name and configure the quick transformation.

3.7.5.4 Edit Quick Transformations

If you have **Control** access to the structure, you can change the associated quick transformations, remove unused transformations, or change the order in which transformations appear in the Quick Transformations panel.

To edit quick transformations:

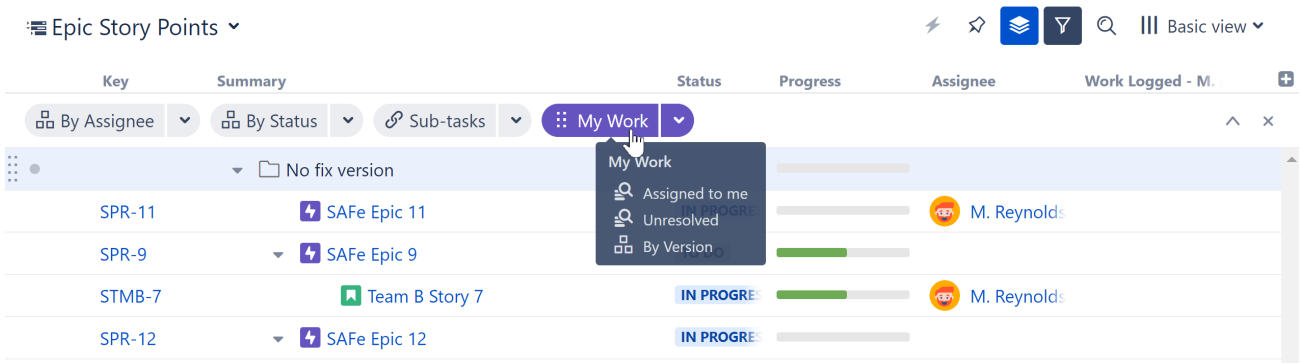
1. To **move a transformation**, drag its icon to the desired position
2. To **toggle, edit,** group(see page 224)**, or delete** a transformation, click the transformation and select the desired action from the drop-down menu



When you edit or delete a default quick transformation, that change only impacts the current structure. Default quick transformations for other structures are not changed.

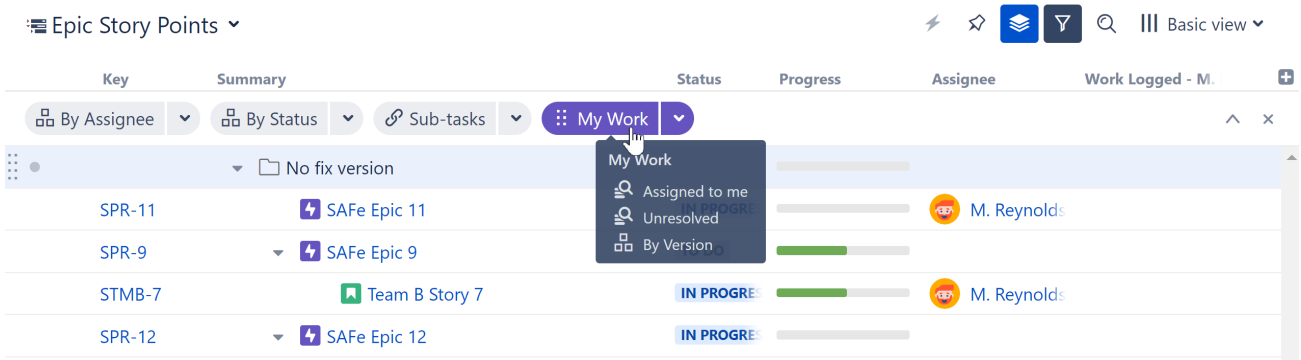
3.7.5.5 Grouping Quick Transformations

If you regularly apply multiple transformations at the same time, you can create a Transformation Group. When you activate the group, all the transformations within it are applied together. Learn more: Transformation Groups(see page 224)



3.7.5.6 Transformation Groups

If you frequently apply multiple Quick Transformations at the same time, you can create a Transformation Group. When you activate the group, all the transformations within it are applied.



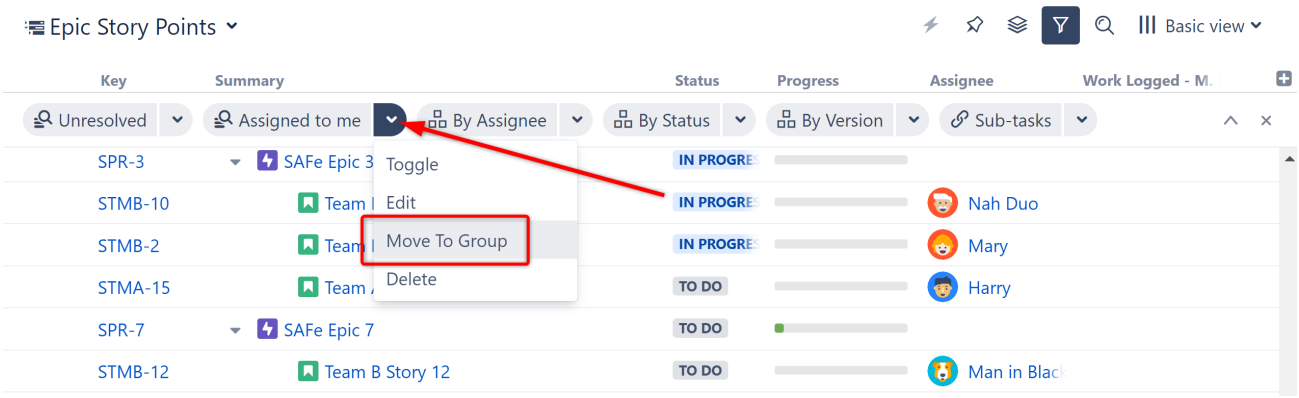
> Grouped transformations are applied in the order they appear within the the group (see Order of Operations(see page 221)).

Learn more:

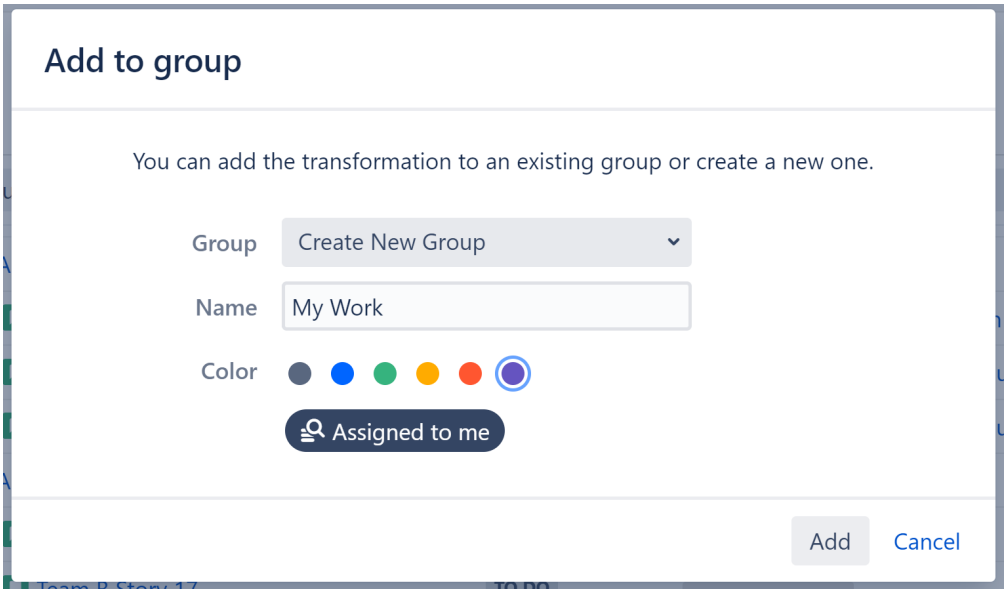
- Creating a Transformation Group(see page 224)
- Editing a Transformation Group(see page 225)
- Editing Transformations within a Group(see page 226)
- Removing Transformations from a Group(see page 227)
- Ungrouping(see page 227)

Creating a Transformation Group

To create a Transformation Group, click the downward arrow next to an existing transformation and select **Move To Group**.



You can add the transformation to an existing group, or create a new one.

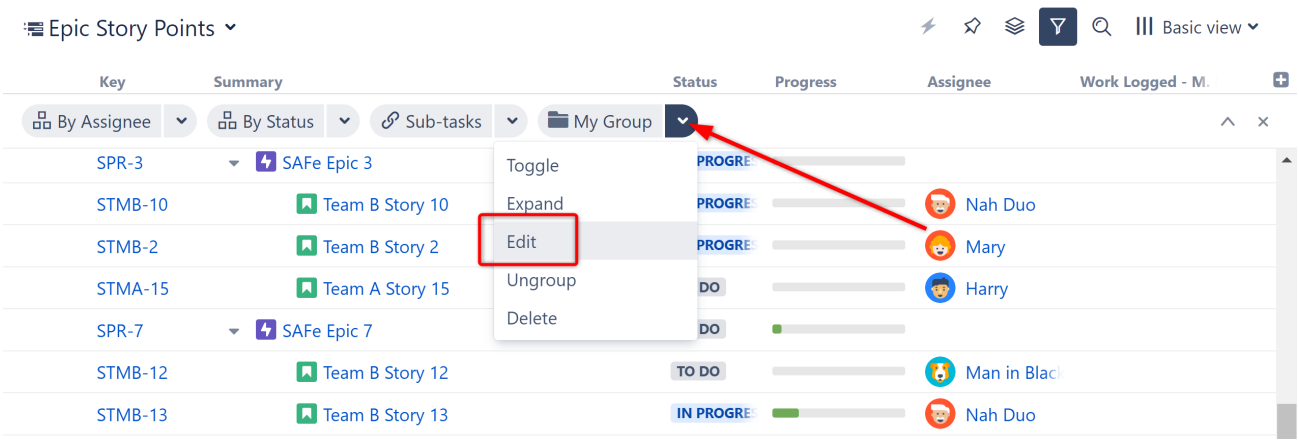


When creating a new group, you can give it a custom name and a unique color to quickly identify it within the Transformations panel.

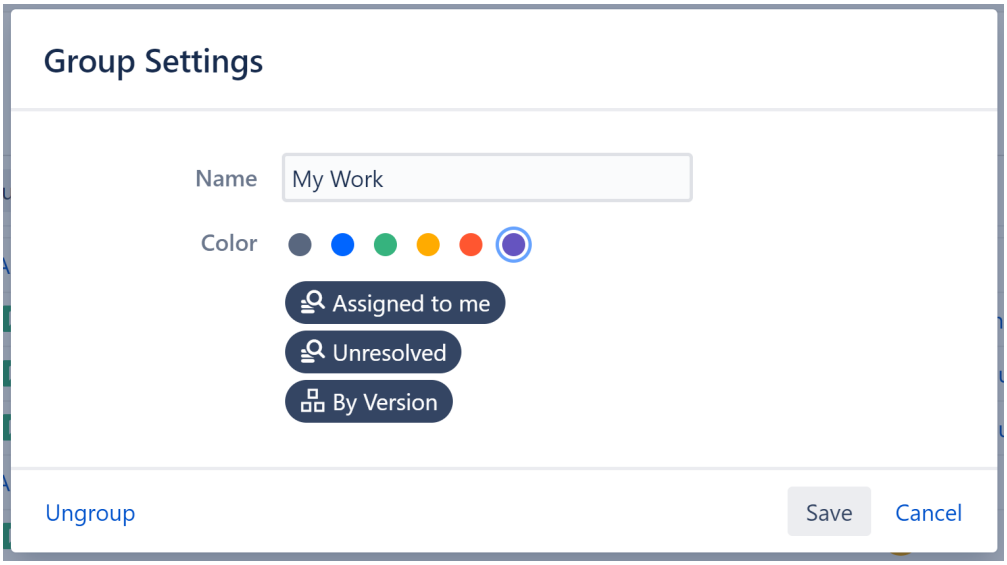
> ⚠ When adding a transformation to an existing group, if either the new transformation or the group is switched off, the group will be switched off once the transformation is added.

Editing a Transformation Group

To edit the properties of an existing group, open the group dropdown menu and select edit.

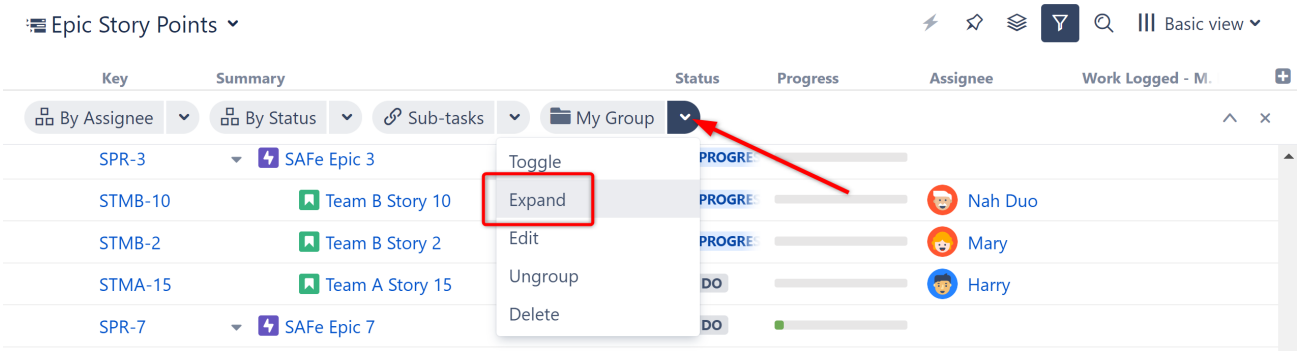


You can edit the group's name and color, or remove or delete any of the transformations within it.



Editing Transformations within a Group

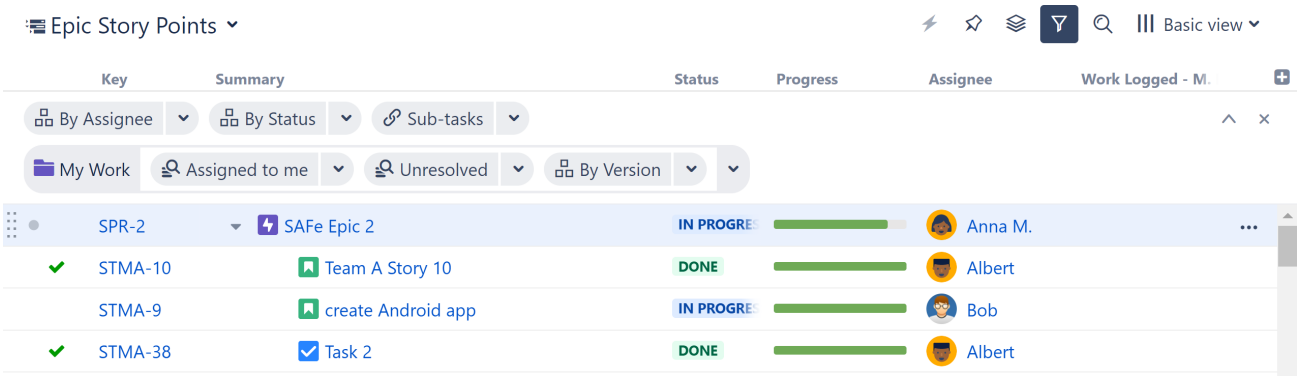
To edit transformations within a group, open the group's menu and select **Expand**.



Once expanded, you can:

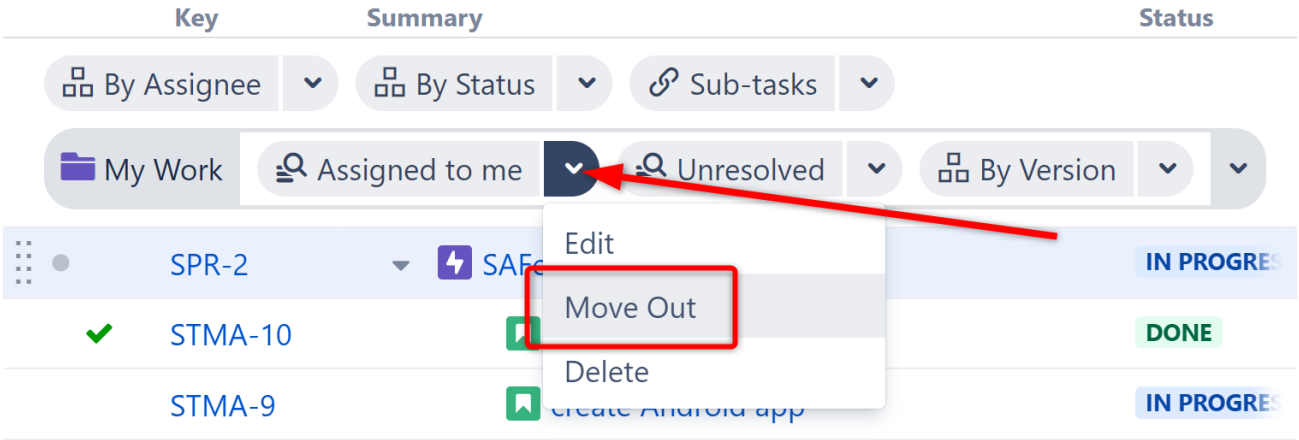
- Edit transformations within the group
- Remove a transformation from the group

- Rearrange transformations to change the order in which they're applied (see Order of Operations(see page 221)).



Removing Transformations from a Group

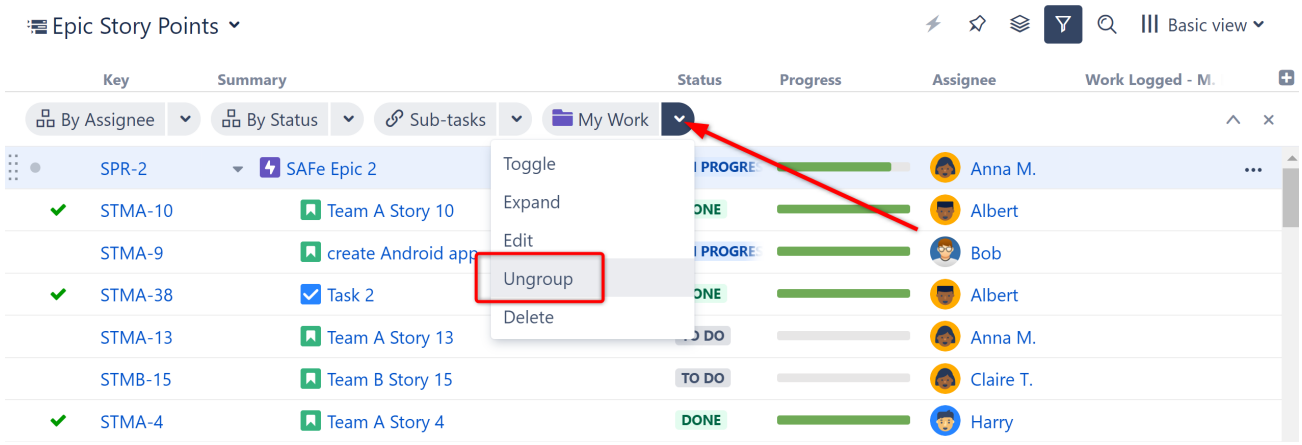
To remove a transformation from a group (and have it appear on the Transformations panel as a single item), expand the group(see page 226), open the transformation's dropdown, and select **Move Out**.



To delete the transformation completely (from the group and from the Transformations panel), select **Delete**.

Ungrouping

To ungroup all the transformations (and make them individual items on the Transformations panel), open the group's dropdown menu and select **Ungroup**.



To delete the group and all the transformations within it, click **Delete**.

3.7.6 Default Quick Transformations

Default quick transformations are shown whenever a structure does not have customized quick transformations, or when displaying query results, clipboard items or other non-structure content.

The following transformations are available:

Transformation	Effect of applying this transformation
Unresolved	Only issues with an empty Resolution field are shown
Assigned to me	Only issues assigned to the current user are shown
By Assignee	All top-level issues are grouped by Assignee
By Status	All top-level issues are grouped by Status
By Version	All top-level issues are grouped by Version
Sub-tasks	Sub-tasks are added to the structure under their parent tasks

Default quick transformations can be edited or deleted.

3.8 Formulas

Formulas can serve a variety of purposes within a structure, including:

- Performing simple or complex calculations based on issue fields, item properties, or other attributes
- Comparing values from multiple fields
- Creating visual notifications, using wiki markup(see page 251)
- Grouping(see page 175), sorting, or filtering(see page 160) items within the structure, using Generators(see page 140)
- Writing data to Jira fields, using Effectors(see page 179)

The following articles will show you how to create your own formulas, customize one of our sample formulas, or simply add a predefined formula(see page 257) to a structure.

- Formula Basics(see page 229)
- Sample Formulas(see page 263)
- Formula Reference Documentation(see page 296)
- Changes to Expr in Structure 7(see page 419)
- Comparision Between Formulas in Cloud and Data Center(see page 427)

3.8.1 Formula Basics

The following articles will walk you through creating simple formulas:

- Creating a Formula(see page 229)
- Mapping Variables(see page 232)
- Expr Language(see page 235)
- Wiki Markup in Formula Columns(see page 251)
- Bundled Formulas(see page 257)
- Saving a Formula(see page 260)
- Opening a Saved Formula(see page 261)
- Debugging a Formula(see page 262)

3.8.1.1 Additional Resources

- Sample Formulas(see page 263) - sample formulas that can be customized to your specific needs
- Formula Reference Documentation(see page 296) - formula and Expr reference docs, including a complete list of available functions

3.8.1.2 Creating a Formula

Formulas can be used within:

- Formula Columns[42] - the expression is calculated for each visible row in the displayed structure or query result.
- Automation[43] - formulas can be used to filter a structure (Filter by Attribute), sort the structure (Sort by Attribute), or group issues (Group by Text Attribute) based on the results of a formula.
- Effectors[44] - the results of a formula can be written to a Jira field using the Attribute to Issue Field Effector[45].

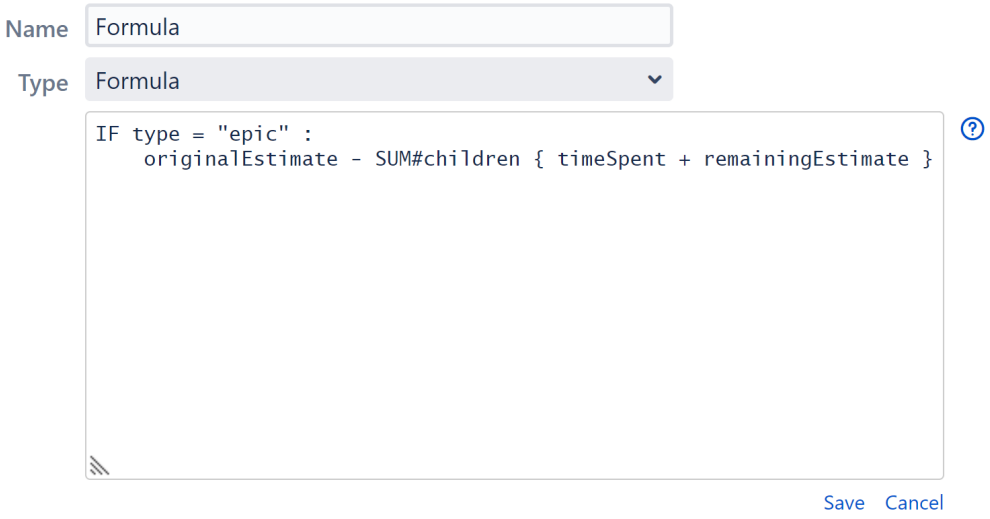
In the following guide, we'll show you had to enter a formula in a formula column. The same steps can also be used within Automation, including Effectors.

1. Enter your formula

Formulas should be constructed using the Expr Language, a simple language that supports variables, arithmetic operations and functions. For a brief overview, see Expr Language(see page 235).

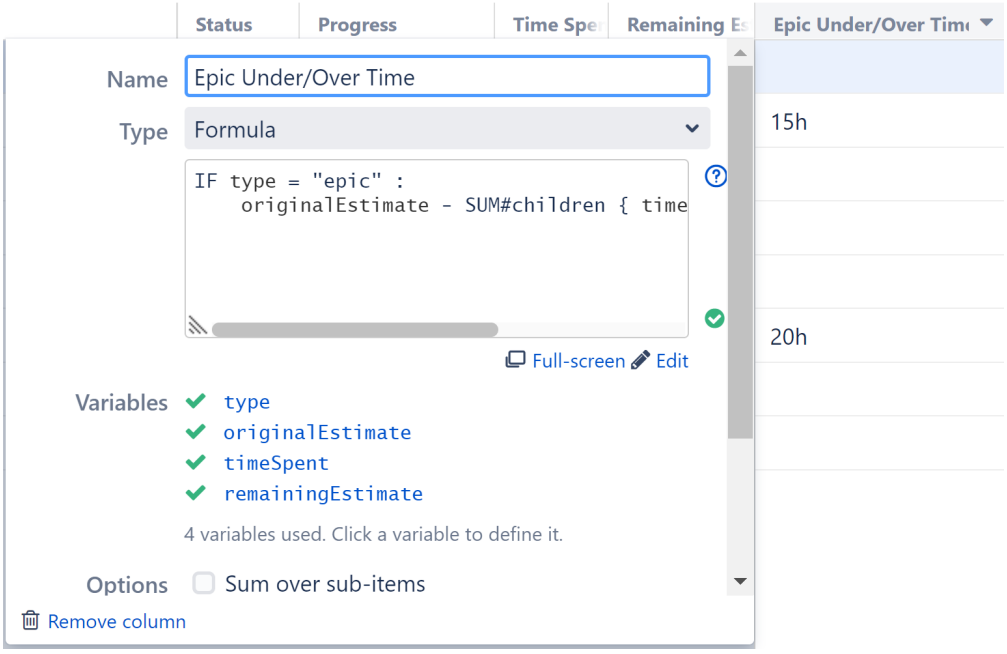
When you're finished, click **Save**.

42 https://wiki.almworks.com/display/structure/Structure+Columns
43 https://wiki.almworks.com/display/structure/Automation
44 https://wiki.almworks.com/display/structure/Effectors
45 https://wiki.almworks.com/display/structure/Attribute+to+Issue+Field+Effector



2. Check your formula

When you click **Save**, Structure will review your formula, attempt to link your variables to issue fields or other attributes and notify you of any errors. If the formula is ready to be used, a green mark is displayed. If it's not, the problematic parts are highlighted in the formula editor with red color.



If there are no issues, you're done! If you're working with a formula column, the results will appear next to the editor.

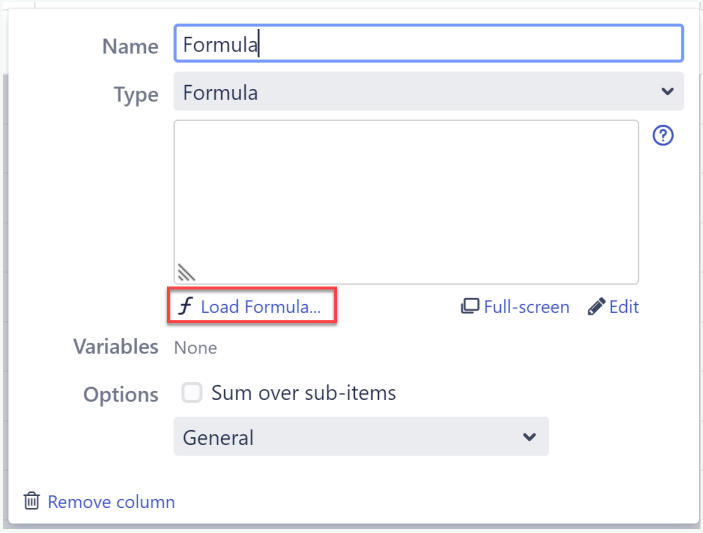
The example above is a simple formula to calculate whether or not we're on target to complete each epic on time:

```
IF type = "epic" :
    originalEstimate - SUM#children { timeSpent + remainingEstimate }
```

In case you're not yet fluent in Expr(see page 235), we're telling Structure to:

1. Check whether the Issue Type is an epic: IF type = "epic" :
2. If so, add the Time Spent and Remaining Estimate for each child issue: { timeSpent + remainingEstimate }
3. Total that value for all the child issues: SUM#children
4. Subtract that total from the epic's Original Estimate: originalEstimate - ...

If you're not sure what formula you want to create, try using one of our bundled formulas(see page 257) - just click **Load Formula...** to see what's available.



Handling Errors

Formula errors are typically due to one of the following:

- **Unmapped Variables** - Structure attempts to map your variables to known attributes; however, depending on how you've named a variable, you may need to map it manually(see page 232).
- **Syntax Error** - occurs when Structure does not understand something in your formula. Often this is something as simple as missing a closing parenthesis or other punctuation. Review the formula and consult our Expr Language(see page 235) guide if you're unsure how to correct it. The part of the formula that failed to parse will be highlighted in red.
- **Function Resolution Error** - occurs when the formula contains an unknown function. Review the formula and make sure the functions marked in red are spelled correctly. If so, check our Expr Function Reference(see page 345) and Aggregate Function Reference(see page 387) to ensure you're using a supported function.

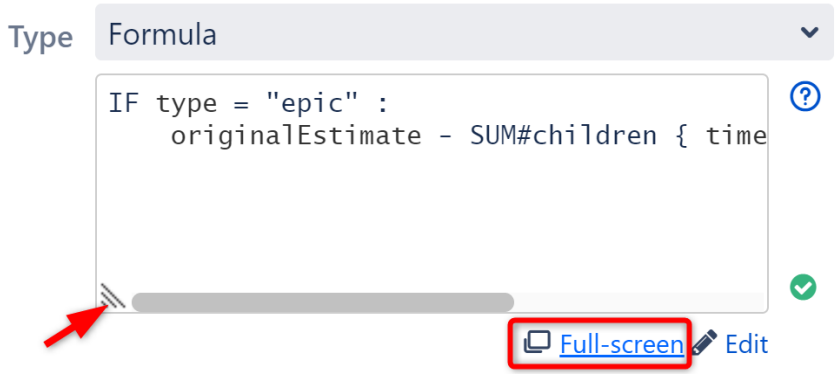
- **Expr Validation Error** - occurs when there is a construct that is not allowed in a formula. Hover over the red highlighted part of the formula to see the error message. If multiple areas are highlighted, there are multiple errors to fix. Check the list of possible validation errors(see page 419) to learn more.

Handling Unexpected Results

In some cases, the formula may pass inspection, but the results aren't what you expected. You may simply need to edit your variables, options or format; or you may need to revise the formula itself by clicking the **Edit** button.

Full Screen

The initial sizing for the formula editor is rather small. If you need more space, drag the corner of the field or click **Full-Screen** to expand the window.



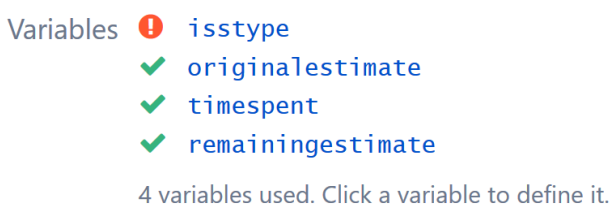
Additional Resources

- Mapping Variables(see page 232)
- Expr Language(see page 235)
- Sample Formulas(see page 263)
- Formula Reference Documentation(see page 296)

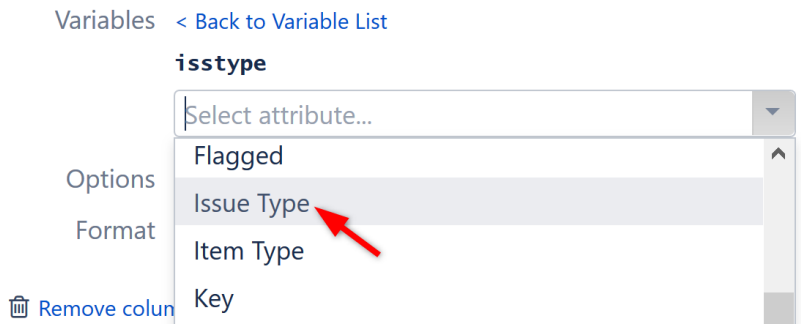
3.8.1.3 Mapping Variables

Most formulas will contain at least one variable (otherwise, the result will be the same for each row in the structure). These variables need to be mapped to *attributes*, which can be issue fields, progress, a hierarchical total, user properties, or another column(see page 233).

As you write your formula, Structure attempts to map your variables to well-known attributes. If Structure is unable to map a variable, it will be marked with a red warning icon, and you will need to map the variable manually.



To map a variable – or to edit an existing mapping – click the variable's name in the variable list or in the formula and select the appropriate attribute from the drop-down list.



Until all variables have been successfully mapped, Structure will use "undefined" as the value of unmapped variables when calculating the formula.

Naming Variables

Structure recognizes most standard and custom fields. The trick is to make sure you're entering their names correctly:

- For one-word names, enter just the field name: **summary** or **priority**
- For multiple-word names: remove all non-letters and replace any spaces with underscores: **story_points**
- Variable names are case insensitive, so story_points and Story_Points will both map to the same field.

For a comprehensive list of recognized variable names, see Standard Variable Reference(see page 402)

Even if Structure successfully maps your variables, it's still a good idea to review them!

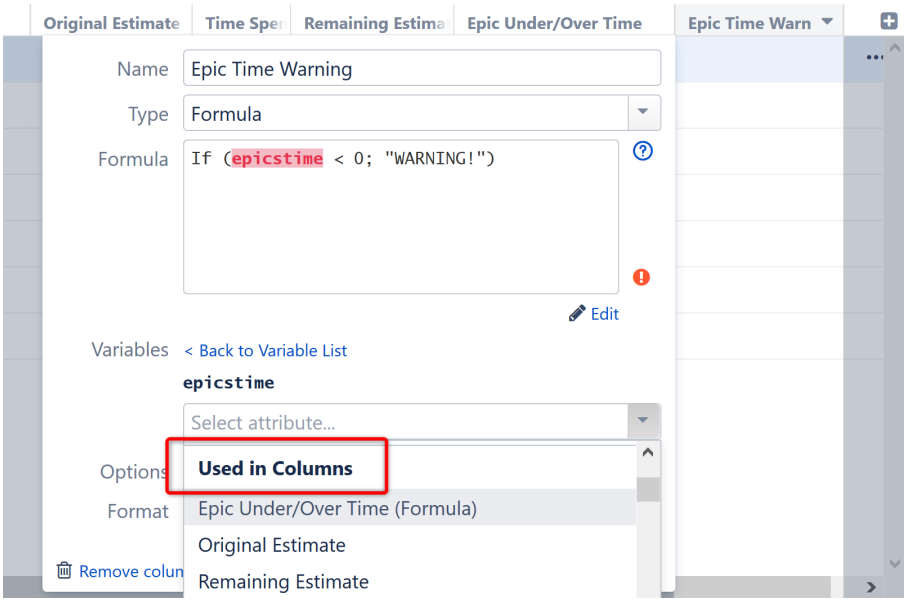
Additional Resources

- Standard Variable Reference(see page 402)
- Columns as Variables(see page 233)

Columns as Variables

When using a formula(see page 229), you can assign a variable to reference another column in the structure, including another formula column.

To assign a variable to another column, look for the **Used in Columns** section of the attribute selection drop-down.

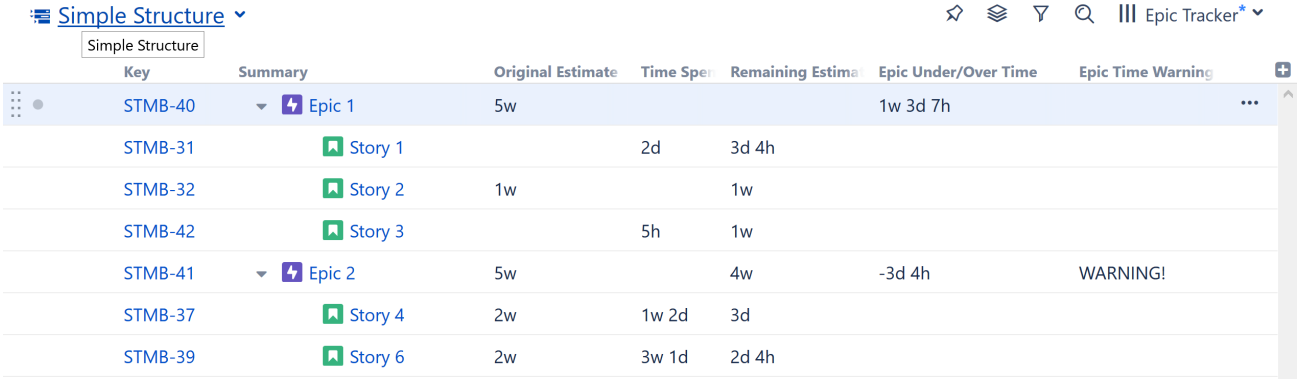


The **Used in Column** attribute copies the existing column. It does not link to that column. This means:

- You can remove the original column without affecting your new formula. The calculations will continue to work just as they did at the moment you first configured the variable.
- If you update the original column and want that update reflected in your new column, you need to reassign the variable.

Example

In the example above, we created a variable to track the results from the **Epics Under/Over Time** formula we created in our Formula Column article(see page 229). If the original formula resulted in a negative value, this column will list a simple "WARNING" text flag. (You could also make this flag more effective with Wiki Markup(see page 251)!)



Key	Summary	Original Estimate	Time Spe	Remaining Estima	Epic Under/Over Time	Epic Time Warning
STMB-40	Epic 1	5w			1w 3d 7h	
STMB-31	Story 1		2d	3d 4h		
STMB-32	Story 2	1w		1w		
STMB-42	Story 3		5h	1w		
STMB-41	Epic 2	5w		4w	-3d 4h	WARNING!
STMB-37	Story 4	2w	1w 2d	3d		
STMB-39	Story 6	2w	3w 1d	2d 4h		

Once we've created our new formula column, if we make changes to the original Epic Under/Over Time column - or even delete it - our Epic Time Warning column will be unaffected. It will still give us a warning whenever we've spent too much time on a particular epic, because the new column (Epic Time Warning) continues to do all the calculations the original column did, even though the original column has changed.

If we want our Warning column updated based on the new changes, we'll need to assign the variable again.

3.8.1.4 Expr Language

Expr Language (pronounced like "expert" without the "t") is a simple language that lets you specify an "expression", or a formula, which is calculated for an issue or another item.

Expr can be used in:

- Formula Columns[46] - the expression is calculated for each visible row in the displayed structure or query result.
- Automation[47] - formulas can be used to filter a structure (Filter by Attribute), sort the structure (Sort by Attribute), or group issues (Group by Text Attribute) based on the results of a formula.
- Effectors[48] - the results of a formula can be written to a Jira field using the Attribute to Issue Field Effector[49].

Expr fundamentals are easy to learn, and yet the language is powerful enough to address complex needs. The following guide will cover the basic requirements of the Expr language.

Language Components

An expression may contain one or more of the following:

- Variables(see page 232) - these are mapped to *attributes*, such as issue fields, custom progress, or a value from another column(see page 233).
- Arithmetic and logical operations - add, subtract, multiply, divide, or compare items.
- Numbers
- Text strings
- Function calls - apply specific calculations to the provided arguments and return a result to be used in the expression.
- Property access - get the value of a particular item property - such as the release date of a fix version.
- Conditional ("IF") expressions - calculate different results based on a specified set of criteria.

There are also more advanced constructs:

- Aggregate Functions - calculate an aggregate (such as the sum or average) of an expression's values calculated for multiple items in the structure.
- Local Variables - introduce a value and reuse it multiple times in the formula.
- User Functions - define a function, or a functional expression, to be reused in the formula or applied to an array of values.
- JQL and S-JQL queries inside a formula - condition the results based on whether an item matches a query.
- Comments - document larger formulas.

Value Types

With Expr you can build formulas that operate on:

- Basic values – numbers and text, which are either a part of the formula, or read from a simple attribute or Jira field, such as an issue's Summary or Story Points.
- Items – values representing a particular object, such as User, Issue, Worklog, Status and others, and typically read from the corresponding Jira field.
- Array values – a sequence of values, allowing you to represent multi-value fields such as Fix Versions, or multiple Entities, in the case of the `worklogs` attribute.

46 https://wiki.almworks.com/display/structure/Structure+Columns
47 https://wiki.almworks.com/display/structure/Automation
48 https://wiki.almworks.com/display/structure/Effectors
49 https://wiki.almworks.com/display/structure/Attribute+to+Issue+Field+Effector

- User function values – representing a piece of formula that is typically applied to each element in an array, for example, for filtering.
- Key-value maps – in rare occasions, these values are produced by the supplied functions like GROUP.
- Undefined – a special value that means "nothing" or "no value".
- Error values – produced if there was a problem calculating a formula.

Normally, you don't need to worry about the value types when writing a formula. The language engine will try to make sense of the formula and convert the values as needed. For more complex formulas, or if something doesn't work as expected, see Expr Function Reference(see page 345) for the expected types for each function.

Variables

Variables are user-defined names, which represent *attributes*, such as:

- Jira issue fields
- Calculated attributes like Progress
- Structure-specific attributes like Item type
- Attributes provided by other Jira apps
- Another formula
- Values from another Structure column

Naming Variables

Variables can contain letters (English only), numbers or underscore ("_") characters. Variables cannot contain spaces, and the first character must be a letter or an underscore.

Examples:

- `priority`
- `sprintName`
- `remaining_estimate`
- `abc11`

As you write your formula, Structure attempts to map your variables to well-known attributes. For example, the "remaining_estimate" variable above will automatically be mapped to the Remaining Estimate field. See Mapping Variables(see page 232) for more information.

Variable names are case-insensitive. `Priority`, `priority` and `pRiOrItY` will all refer to the **same** variable.

Local Variables

Local variables are helpful when an expression needs to be used in the same formula several times. For example:

```
IF time_spent + remaining_estimate > 0 :
   time_spent / (time_spent + remaining_estimate)
```

You can see that in this formula we are using "`time_spent + remaining_estimate`" twice – once when we check that it's not zero (so we don't divide by zero) and again when we divide by it.

Instead of repeating the expression every time, we can rewrite this formula using the **`WITH`** construct:

```
WITH total_time = time_spent + remaining_estimate :
  IF total_time > 0 :
     time_spent / total_time
```

You can define multiple local variables in succession. You can also use previously defined local variables when defining additional local variables. For example:

```
WITH total_time = time_spent + remaining_estimate :
WITH progress = (IF total_time > 0 : time_spent / total_time) :
  IF(progress > 0.5, "Great Progress!", progress > 0.2, "Good Progress", "Needs
Progress!")
```

Note the position of the colon (":") – it must be present where each local variable definition ends.

Functions

A function calculates a value based on its arguments and, sometimes, some external aspect. A function call is written as the function name, followed by parentheses, which may or may not contain arguments.

Examples:

- `SUM(-original_estimate, remaining_estimate, time_spent)`
- `CASE(priority, 'High*', 5, 1)`
- `TODAY()`

Function names are case-insensitive. You can write `TODAY()` or `Today()`.

There are 100+ standard functions available with Structure – see Expr Function Reference(see page 345) for a complete list.

Function arguments may be separated by comma (,) or semicolon (;). But in every function call within a formula, you need to use either all commas or all semicolons.

Chained Function Calls

The chained notation allows you to easily apply a sequence of functions to a value, simply by listing each function one after the other, separated by a (.) dot.

- Standard notation: `F3(F2(F1(x)))`
- Chain notation: `x.F1().F2().F3()`

When you use the chain notation, the value that comes before the dot becomes the first argument for the function. If the function takes multiple arguments, the rest of the arguments must be written in parentheses.

For example:

```
created.FORMAT_DATETIME("yyyy").CONCAT(" year issue")
```

In this example, FORMAT_DATETIME takes the date value in "created" and formats it based on the argument in parenthesis ("yyyy"). CONCAT takes the result from FORMAT_DATETIME and joins it with " year issue".

Aggregate Functions

An aggregate function calculates some aggregate value (like sum or minimum) based on the values in a number of rows, typically for all sub-issues. Aggregate functions are written very similar to standard functions, except they use curly braces: `SUM{x}`.

Examples:

- `SUM { remaining_estimate + time_spent }` – calculates the total effort (estimated and actual) for the issue and all its sub-issues.
- `MAX { resolved_date - created_date }` – calculates the maximum time it took to resolve an issue, among the issue and its sub-issues.

They can also contain **modifiers**, which influence how the aggregation works:

- `SUM#all { business_value }` – this will force the function to include values from all duplicate items in the total. (By default, duplicates are ignored.)

See Aggregate Function Reference(see page 387) for a complete list of available aggregate functions and modifiers.

> ⚠ Any local variables(see page 244) used inside an aggregate function must also be declared inside the function - within the { }.

User Functions

A user function allows you to define a locally-used function within a formula. User functions can be defined in a similar manner as local variables(see page 244):

```
WITH square(x) = x * x :
  square(impactField) / square(storyPoints)
```

In this example, the user function is given a name ("square") and then used to perform the same calculation on multiple fields. To learn more, see the language reference(see page 0).

User Functions for Arrays - using the "$" character

When you need to perform an operation on each element in an array, you can use a user function such as the one above, or simplify it using “$” to indicate each element in the array.

```
worklogs.FILTER($.author = ME())
```

In this example, the "$" tells Structure to apply "author = ME()" to each element in worklogs - if the author is the current user, it returns true and that worklog will be included in the FILTER results.

This method becomes very powerful when you combine multiple user functions together. To learn more, see the language reference(see page 316).

Embedded Queries (JQL and S-JQL)

You can embed JQL and Structured JQL(see page 428) queries inside an Expr formula, using a construct similar to Aggregate Functions(see page 247). The result will be a boolean value:

- `1` (true) if the current row matches the query
- `0` (false) otherwise

For example:

```
// Collect total story points from all sub-issues assigned to members of Team2 group,
unless the stories are under folder "Special"
SUM {
    IF JQL { assignee in membersOf("Team2") } :
    IF NOT SJQL { descendant of folder("Special") } :
      storyPoints
}
```

Since JQL is a Jira-based query, it will work only on issues; the result will be `0` on other types of items. S-JQL can be used for more complex queries applicable to the whole structure.

Numbers

Formulas support whole numbers, decimals, or fractions. Commas, spaces, locale-specific, percentage, currency or scientific formats are not supported.

Recognized as a number	Not recognized as a number
`0`	`0,0`
`1000`	`1,000`
`1234567890123456`	`1 100 025`
`11.25`	`1.234e+04`
`.111`	`($100)`

You can write a number that is written with a locale-specific decimal and thousands separator as a text value, and it will be automatically converted to a number if needed. For example:

- `"1 122,25" * 2 → 2244.5`

Texts

Text Strings

Text strings are a sequence of characters enclosed either in single (') or double quotes ("). Examples:

- `'a text in single quotes may contain " (a double quote)'`
- `"a text in double quotes may contain ' (a single quote)"`
- `""`

Everything within a text string is retained exactly when the expression is evaluated or displayed, except for the following:

- A sequence of two backslashes (`\\`) is converted to a single backslash (`\`).
- A sequence of a backslash and a single quote (`\'`) is converted to a single quote character (`'`) for text values enclosed in single quotes.

- A sequence of a backslash and a double quote (\") is converted to a double quote character (") for the text values enclosed in double quotes.

Text Snippets

Text Snippets allow you to generate strings using variables and expressions. This is particularly helpful in formulas that utilize wiki markup(see page 251).When using text snippets:

- The snippet should be enclosed with """ (three double quotes, at the beginning and at the end)
- The expression portion of the snippet is introduced using the '$' symbol and should be enclosed in braces { }

```
""" $var1 + $var2 = ${var1 + var2} """

""" this $glass is half-${IF optimist: 'full' ELSE: 'empty'} """
```

Operations

Expr provides basic arithmetic operations, comparisons, text operations and logical operations.

Operations	Comments
`+ - * /`	Basic operators. When used, the value is converted to a number. Follows the general precedence rules for arithmetic, so (2 + 3 * 4 = 14).
`= !=`	Equality and non-equality: if either part of the comparison is a number, the other part is also converted into a number. If both values are texts, then text comparison is used. Text comparison ignores leading and trailing whitespace and is case-insensitive (according to Jira's system locale).
`< <= > >=`	Numerical comparisons. When used, both values are converted to numbers.
`AND`, `OR`, `NOT`	Logical operations.
`CONCAT`	An operation that joins together two text strings. Works similar to the function of the same name: `a CONCAT b` is the same as `CONCAT(a, b)`.
`( )`	Parentheses can be used to group the results of operations prior to passing them to other operations.

Order of Operations

When several types of operations are used, they are done in the following order:

1. Arithmetic operations
2. Text operations (CONCAT)
3. Comparison operations
4. Logical operations.

For detailed specification, see Expr Advanced Reference(see page 296).

Property Access

Formulas can get the value of an item's property using the following notation: `object.property`

```
fixVersion.releaseDate  //returns the release date for the fixVersion
```

You can also string multiple property calls together:

```
project.lead.emailAddress  //returns the email address of the lead for the project
```

For a complete list of supported properties, see Item Property Reference(see page 406).

Conditional Expressions

Simple "IF" expressions can be declared using the IF() function(see page 345), but for more elaborate IF cases, with multiple conditions and/or requiring an ELSE option, a conditional expression can be used.

```
WITH total = x + y:
  IF total > 0:
     x / total
  ELSE : error
```

Note: the ":" after "ELSE" is optional – in the example above, we've included it for readability.

Comments

Comments are helpful when you have a large formula or when a reader might need explanations of what is being calculated. It's a good idea to add comments wherever the formula is not trivial.

- To add a single line of comment, begin the comment with `//`
- To add multiple lines of comment, start the comment with `/*` and end the comment with `*/`

Example:

```
// This is a single-line comment.

/* This is a multi-line comment.
   It can be useful for longer explanations. */
```

Additional Resources

- Formula Reference Documentation(see page 296)
- Sample Formulas(see page 263)
- Wiki Markup in Formula Columns(see page 251)

For a more in-depth study, see our Expr Advanced Reference(see page 296) and Formula Reference Documentation(see page 296).

You can view examples of Expr formulas in Sample Formulas(see page 263) or by adding bundled formulas(see page 257) to your structure.To see the underlying formula, simply open the column options(see page 501) panel.

Expr Language - Language Components

An expression may contain one or more of the following:

- Variables(see page 232) - these are mapped to *attributes*, such as issue fields, custom progress, or a value from another column(see page 233).
- Arithmetic and logical operations - add, subtract, multiply, divide, or compare items.
- Numbers
- Text strings
- Function calls - apply specific calculations to the provided arguments and return a result to be used in the expression.
- Property access - get the value of a particular item property - such as the release date of a fix version.
- Conditional ("IF") expressions - calculate different results based on a specified set of criteria.

There are also more advanced constructs:

- Aggregate Functions - calculate an aggregate (such as the sum or average) of an expression's values calculated for multiple items in the structure.
- Local Variables - introduce a value and reuse it multiple times in the formula.
- User Functions - define a function, or a functional expression, to be reused in the formula or applied to an array of values.
- JQL and S-JQL queries inside a formula - condition the results based on whether an item matches a query.
- Comments - document larger formulas.

Expr Language - Value Types

With Expr you can build formulas that operate on:

- Basic values – numbers and text, which are either a part of the formula, or read from a simple attribute or Jira field, such as an issue's Summary or Story Points.
- Items – values representing a particular object, such as User, Issue, Worklog, Status and others, and typically read from the corresponding Jira field.
- Array values – a sequence of values, allowing you to represent multi-value fields such as Fix Versions, or multiple Entities, in the case of the `worklogs` attribute.
- User function values – representing a piece of formula that is typically applied to each element in an array, for example, for filtering.
- Key-value maps – in rare occasions, these values are produced by the supplied functions like GROUP.
- Undefined – a special value that means "nothing" or "no value".
- Error values – produced if there was a problem calculating a formula.

Normally, you don't need to worry about the value types when writing a formula. The language engine will try to make sense of the formula and convert the values as needed. For more complex formulas, or if something doesn't work as expected, see Expr Function Reference(see page 345) for the expected types for each function.

Expr Language - Variables

Variables are user-defined names, which represent *attributes*, such as:

- Jira issue fields
- Calculated attributes like Progress
- Structure-specific attributes like Item type
- Attributes provided by other Jira apps
- Another formula
- Values from another Structure column

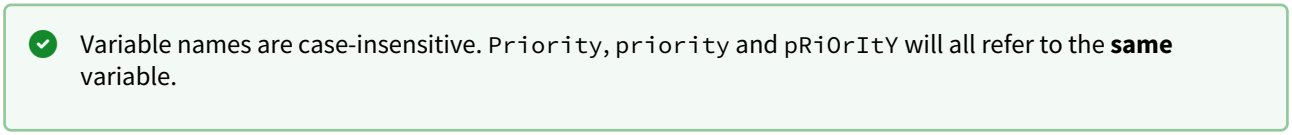
Naming Variables

Variables can contain letters (English only), numbers or underscore ("_") characters. Variables cannot contain spaces, and the first character must be a letter or an underscore.

Examples:

- `priority`
- `sprintName`
- `remaining_estimate`
- `abc11`

As you write your formula, Structure attempts to map your variables to well-known attributes. For example, the "remaining_estimate" variable above will automatically be mapped to the Remaining Estimate field. See Mapping Variables(see page 232) for more information.


Variable names are case-insensitive. `Priority`, `priority` and `pRiOrItY` will all refer to the **same** variable.

Local Variables

Local variables are helpful when an expression needs to be used in the same formula several times. For example:

```
IF time_spent + remaining_estimate > 0 :
   time_spent / (time_spent + remaining_estimate)
```

You can see that in this formula we are using "`time_spent + remaining_estimate`" twice – once when we check that it's not zero (so we don't divide by zero) and again when we divide by it.

Instead of repeating the expression every time, we can rewrite this formula using the **`WITH`** construct:

```
WITH total_time = time_spent + remaining_estimate :
  IF total_time > 0 :
     time_spent / total_time
```

You can define multiple local variables in succession. You can also use previously defined local variables when defining additional local variables. For example:

```
WITH total_time = time_spent + remaining_estimate :
WITH progress = (IF total_time > 0 : time_spent / total_time) :
  IF(progress > 0.5, "Great Progress!", progress > 0.2, "Good Progress", "Needs
Progress!")
```

Note the position of the colon (":") – it must be present where each local variable definition ends.

Expr Language - Local Variables

Local variables are helpful when an expression needs to be used in the same formula several times. For example:

```
IF time_spent + remaining_estimate > 0 :
   time_spent / (time_spent + remaining_estimate)
```

You can see that in this formula we are using "`time_spent + remaining_estimate`" twice – once when we check that it's not zero (so we don't divide by zero) and again when we divide by it.

Instead of repeating the expression every time, we can rewrite this formula using the **`WITH`** construct:

```
WITH total_time = time_spent + remaining_estimate :
  IF total_time > 0 :
     time_spent / total_time
```

You can define multiple local variables in succession. You can also use previously defined local variables when defining additional local variables. For example:

```
WITH total_time = time_spent + remaining_estimate :
WITH progress = (IF total_time > 0 : time_spent / total_time) :
  IF(progress > 0.5, "Great Progress!", progress > 0.2, "Good Progress", "Needs
Progress!")
```

Note the position of the colon (":") – it must be present where each local variable definition ends.

Expr Language - Functions

A function calculates a value based on its arguments and, sometimes, some external aspect. A function call is written as the function name, followed by parentheses, which may or may not contain arguments.

Examples:

- `SUM(-original_estimate, remaining_estimate, time_spent)`
- `CASE(priority, 'High*', 5, 1)`
- `TODAY()`

Function names are case-insensitive. You can write `TODAY()` or `Today()`.

There are 100+ standard functions available with Structure – see Expr Function Reference(see page 345) for a complete list.

> Function arguments may be separated by comma (,) or semicolon (;). But in every function call within a formula, you need to use either all commas or all semicolons.

Chained Function Calls

The chained notation allows you to easily apply a sequence of functions to a value, simply by listing each function one after the other, separated by a (.) dot.

- Standard notation: `F3(F2(F1(x)))`
- Chain notation: `x.F1().F2().F3()`

When you use the chain notation, the value that comes before the dot becomes the first argument for the function. If the function takes multiple arguments, the rest of the arguments must be written in parentheses.

For example:

```
created.FORMAT_DATETIME("yyyy").CONCAT(" year issue")
```

In this example, FORMAT_DATETIME takes the date value in "created" and formats it based on the argument in parenthesis ("yyyy"). CONCAT takes the result from FORMAT_DATETIME and joins it with " year issue".

Aggregate Functions

An aggregate function calculates some aggregate value (like sum or minimum) based on the values in a number of rows, typically for all sub-issues. Aggregate functions are written very similar to standard functions, except they use curly braces: `SUM{x}`.

Examples:

- `SUM { remaining_estimate + time_spent }` – calculates the total effort (estimated and actual) for the issue and all its sub-issues.
- `MAX { resolved_date - created_date }` – calculates the maximum time it took to resolve an issue, among the issue and its sub-issues.

They can also contain **modifiers**, which influence how the aggregation works:

- `SUM#all { business_value }` – this will force the function to include values from all duplicate items in the total. (By default, duplicates are ignored.)

See Aggregate Function Reference(see page 387) for a complete list of available aggregate functions and modifiers.

> Any local variables(see page 244) used inside an aggregate function must also be declared inside the function - within the { } .

User Functions

A user function allows you to define a locally-used function within a formula. User functions can be defined in a similar manner as local variables(see page 244):

```
WITH square(x) = x * x :
  square(impactField) / square(storyPoints)
```

In this example, the user function is given a name ("square") and then used to perform the same calculation on multiple fields. To learn more, see the language reference(see page 0).

User Functions for Arrays - using the "$" character

When you need to perform an operation on each element in an array, you can use a user function such as the one above, or simplify it using “$” to indicate each element in the array.

```
worklogs.FILTER($.author = ME())
```

In this example, the "$" tells Structure to apply "author = ME()" to each element in worklogs - if the author is the current user, it returns true and that worklog will be included in the FILTER results.

This method becomes very powerful when you combine multiple user functions together. To learn more, see the language reference(see page 316).

Embedded Queries (JQL and S-JQL)

You can embed JQL and Structured JQL(see page 428) queries inside an Expr formula, using a construct similar to Aggregate Functions(see page 247). The result will be a boolean value:

- `1` (true) if the current row matches the query
- `0` (false) otherwise

For example:

```
// Collect total story points from all sub-issues assigned to members of Team2 group,
unless the stories are under folder "Special"
SUM {
    IF JQL { assignee in membersOf("Team2") } :
    IF NOT SJQL { descendant of folder("Special") } :
      storyPoints
}
```

Since JQL is a Jira-based query, it will work only on issues; the result will be `0` on other types of items. S-JQL can be used for more complex queries applicable to the whole structure.

Expr Language - Chained Function Calls

The chained notation allows you to easily apply a sequence of functions to a value, simply by listing each function one after the other, separated by a (.) dot.

- Standard notation: `F3(F2(F1(x)))`
- Chain notation: `x.F1().F2().F3()`

When you use the chain notation, the value that comes before the dot becomes the first argument for the function. If the function takes multiple arguments, the rest of the arguments must be written in parentheses.

For example:

```
created.FORMAT_DATETIME("yyyy").CONCAT(" year issue")
```

In this example, FORMAT_DATETIME takes the date value in "created" and formats it based on the argument in parenthesis ("yyyy"). CONCAT takes the result from FORMAT_DATETIME and joins it with " year issue".

Expr Language - Aggregate Functions

An aggregate function calculates some aggregate value (like sum or minimum) based on the values in a number of rows, typically for all sub-issues. Aggregate functions are written very similar to standard functions, except they use curly braces: `SUM{x}`.

Examples:

- `SUM { remaining_estimate + time_spent }` – calculates the total effort (estimated and actual) for the issue and all its sub-issues.
- `MAX { resolved_date - created_date }` – calculates the maximum time it took to resolve an issue, among the issue and its sub-issues.

They can also contain **modifiers**, which influence how the aggregation works:

- `SUM#all { business_value }` – this will force the function to include values from all duplicate items in the total. (By default, duplicates are ignored.)

See Aggregate Function Reference(see page 387) for a complete list of available aggregate functions and modifiers.

> ⚠ Any local variables(see page 244) used inside an aggregate function must also be declared inside the function - within the { }.

Expr Language - User Functions

A user function allows you to define a locally-used function within a formula. User functions can be defined in a similar manner as local variables(see page 244):

```
WITH square(x) = x * x :
  square(impactField) / square(storyPoints)
```

In this example, the user function is given a name ("square") and then used to perform the same calculation on multiple fields. To learn more, see the language reference(see page 0).

User Functions for Arrays - using the "$" character

When you need to perform an operation on each element in an array, you can use a user function such as the one above, or simplify it using “$” to indicate each element in the array.

```
worklogs.FILTER($.author = ME())
```

In this example, the "$" tells Structure to apply "author = ME()" to each element in worklogs - if the author is the current user, it returns true and that worklog will be included in the FILTER results.

This method becomes very powerful when you combine multiple user functions together. To learn more, see the language reference(see page 316).

Embedded Queries (JQL and S-JQL)
You can embed JQL and Structured JQL(see page 428) queries inside an Expr formula, using a construct similar to Aggregate Functions(see page 247). The result will be a boolean value:

- `1` (true) if the current row matches the query
- `0` (false) otherwise

For example:

```
// Collect total story points from all sub-issues assigned to members of Team2 group,
unless the stories are under folder "Special"
SUM {
    IF JQL { assignee in membersOf("Team2") } :
    IF NOT SJQL { descendant of folder("Special") } :
      storyPoints
}
```

Since JQL is a Jira-based query, it will work only on issues; the result will be `0` on other types of items. S-JQL can be used for more complex queries applicable to the whole structure.

Expr Language - Numbers

Formulas support whole numbers, decimals, or fractions. Commas, spaces, locale-specific, percentage, currency or scientific formats are not supported.

Recognized as a number	Not recognized as a number
`0`	`0,0`
`1000`	`1,000`
`1234567890123456`	`1 100 025`
`11.25`	`1.234e+04`
`.111`	`($100)`

You can write a number that is written with a locale-specific decimal and thousands separator as a text value, and it will be automatically converted to a number if needed. For example:

- `"1 122,25" * 2 → 2244.5`

Expr Language - Texts

Text Strings

Text strings are a sequence of characters enclosed either in single (') or double quotes ("). Examples:

- `'a text in single quotes may contain " (a double quote)'`
- `"a text in double quotes may contain ' (a single quote)"`
- `""`

Everything within a text string is retained exactly when the expression is evaluated or displayed, except for the following:

- A sequence of two backslashes (`\\`) is converted to a single backslash (`\`).
- A sequence of a backslash and a single quote (`\'`) is converted to a single quote character (`'`) for text values enclosed in single quotes.
- A sequence of a backslash and a double quote (`\"`) is converted to a double quote character (") for the text values enclosed in double quotes.

Text Snippets

Text Snippets allow you to generate strings using variables and expressions. This is particularly helpful in formulas that utilize wiki markup(see page 251).When using text snippets:

- The snippet should be enclosed with """ (three double quotes, at the beginning and at the end)
- The expression portion of the snippet is introduced using the '$' symbol and should be enclosed in braces { }

```
""" $var1 + $var2 = ${var1 + var2} """

""" this $glass is half-${IF optimist: 'full' ELSE: 'empty'} """
```

Expr Language - Operations

Expr provides basic arithmetic operations, comparisons, text operations and logical operations.

Operations	Comments
`+ - * /`	Basic operators. When used, the value is converted to a number. Follows the general precedence rules for arithmetic, so (2 + 3 * 4 = 14).
`= !=`	Equality and non-equality: if either part of the comparison is a number, the other part is also converted into a number. If both values are texts, then text comparison is used. Text comparison ignores leading and trailing whitespace and is case-insensitive (according to Jira's system locale).
`< <= > >=`	Numerical comparisons. When used, both values are converted to numbers.
`AND, OR, NOT`	Logical operations.
`CONCAT`	An operation that joins together two text strings. Works similar to the function of the same name: `a CONCAT b` is the same as `CONCAT(a, b)`.

Operations	Comments
( )	Parentheses can be used to group the results of operations prior to passing them to other operations.

Order of Operations

When several types of operations are used, they are done in the following order:

1. Arithmetic operations
2. Text operations (CONCAT)
3. Comparison operations
4. Logical operations.

For detailed specification, see Expr Advanced Reference(see page 296).

Expr Language - Property Access

Formulas can get the value of an item's property using the following notation: `object.property`

```
fixVersion.releaseDate  //returns the release date for the fixVersion
```

You can also string multiple property calls together:

```
project.lead.emailAddress  //returns the email address of the lead for the project
```

For a complete list of supported properties, see Item Property Reference(see page 406).

Expr Language - Conditional Expressions

Simple "IF" expressions can be declared using the IF() function(see page 345), but for more elaborate IF cases, with multiple conditions and/or requiring an ELSE option, a conditional expression can be used.

```
WITH total = x + y:
  IF total > 0:
     x / total
  ELSE : error
```

Note: the ":" after "ELSE" is optional – in the example above, we've included it for readability.

Expr Language - Comments

Comments are helpful when you have a large formula or when a reader might need explanations of what is being calculated. It's a good idea to add comments wherever the formula is not trivial.

- To add a single line of comment, begin the comment with `//`
- To add multiple lines of comment, start the comment with `/*` and end the comment with `*/`

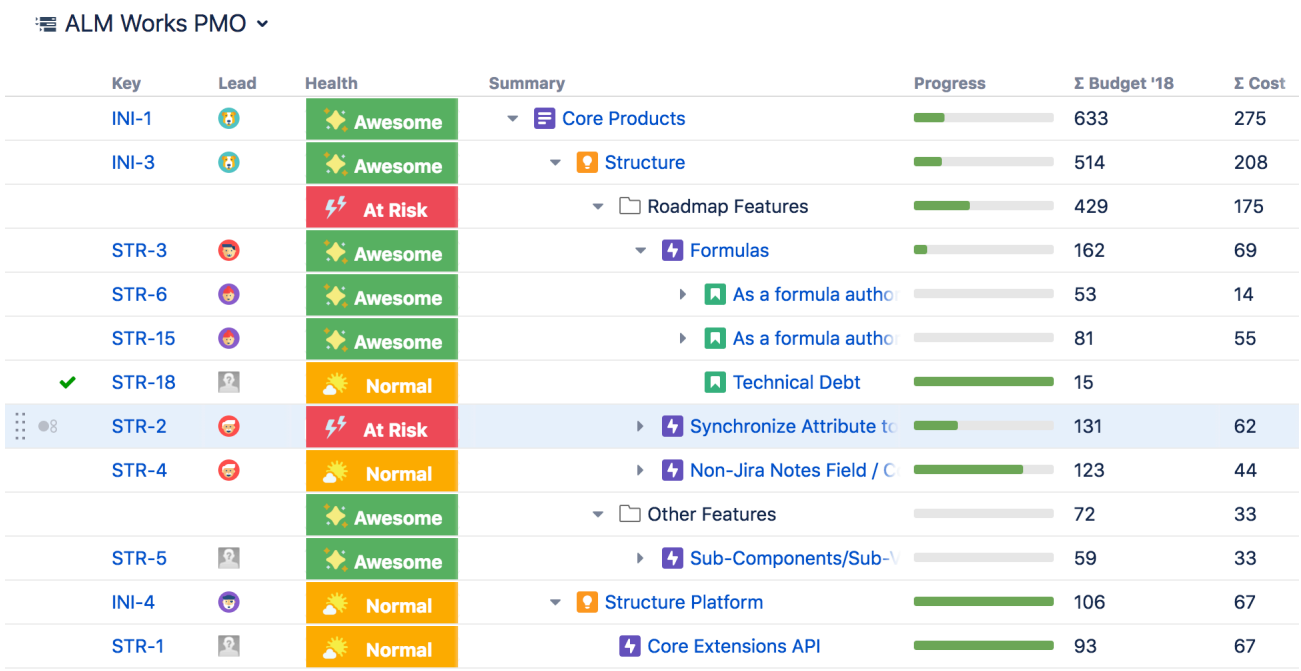
Example:

```
// This is a single-line comment.

/* This is a multi-line comment.
   It can be useful for longer explanations. */
```

3.8.1.5 Wiki Markup in Formula Columns

> The following article only applies to Structure for Jira Server or Data Center. If you're using Structure for Jira Cloud, see Markdown in Formula Columns[50].

By adding wiki markup to a formula column(see page 463), you can call attention to critical information, color-code data fields, or add other visual customizations to a structure.



Wiki markup allows you to:

- Specify the text color within a column
- Highlight cells with background coloring
- Insert images
- Add emojis

50 https://wiki.almworks.com/display/structure/.Markdown+in+Formula+Columns+v8.3

Sorry, the widget is not supported in this export.
But you can reach it using the following URL:

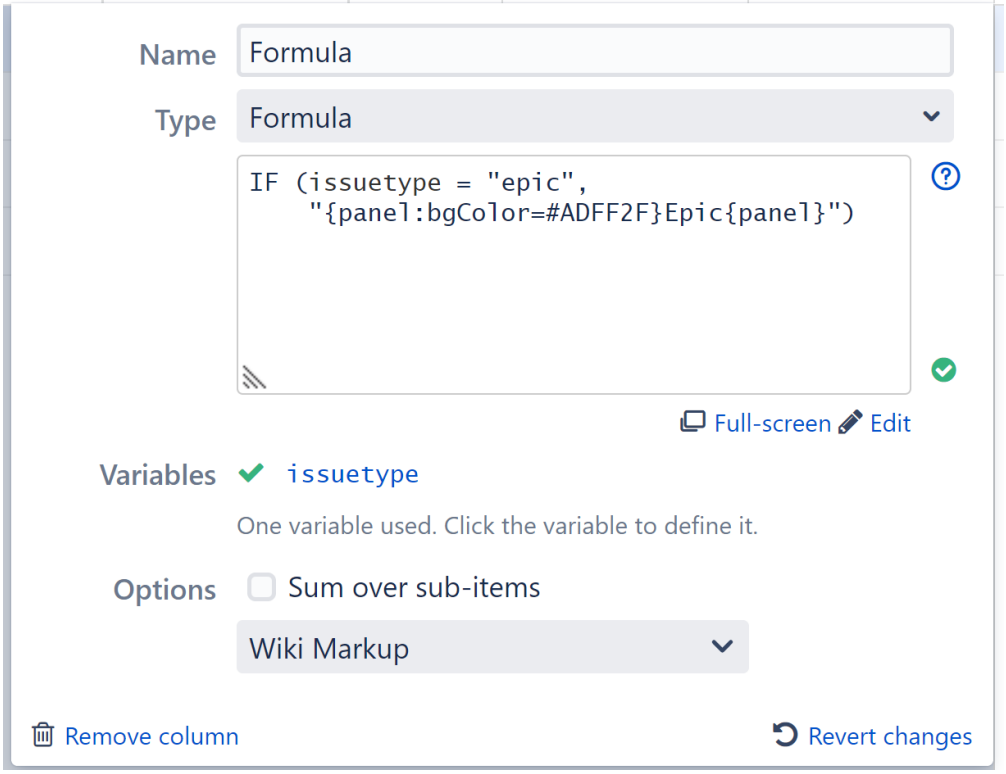
https://www.youtube.com/watch?v=7boTHUDge0c

Using Wiki Markup

To add wiki markup to a formula column:

Click the Add Column button (**+**) and select **Formula**

1. Include wiki markup language in your formula column, surrounded by double quotes ("). See Markup Options(see page 253) below for more details.
2. Under the Format menu, select Wiki Markup. (This is important - your content will not display correctly unless the Wiki Markup format is selected.)



As you save/update the formula, your new column should update automatically. Once you're finished, click anywhere on your structure to close the Add Column dialogue and see your new column.

The example above highlights all the Epics in the structure:

Here's the formula we used - just in case you want to try it yourself:

```
IF(issuetype="Epic", "{panel:bgColor=#ADFF2F}Epic{panel}")
```

With a few more If statements, you could color-code your entire structure by issue type. Or you could assign different colors to each Assignee or some other custom field. The possibilities are endless!

Markup Options

Structure uses the Jira Markup language to enable wiki markup within formula columns.

Using wiki markup, you can add the following elements to a cell:

- Custom text formatting
- Text, background and border color
- Images
- Emojis

You can find a complete list of available formatting options and conventions on Jira's Text Formatting Notation Help page[51].

While it is possible to add tables and lists to a formula column, we do not recommend it. Due to the limited space, these items may not appear as expected.

Export

Wiki Markup can be exported to Excel or printed, using Structure's Export feature.

Your markup should export just as it appears in Structure, with some exceptions:

- Colored borders are not exported to Excel or printable.
- When exporting to Excel, text cannot be combined with emojis or other images within the same cell. If both are present, only the text will be exported.

51 https://jira.atlassian.com/secure/WikiRendererHelpAction.jspa?section=all

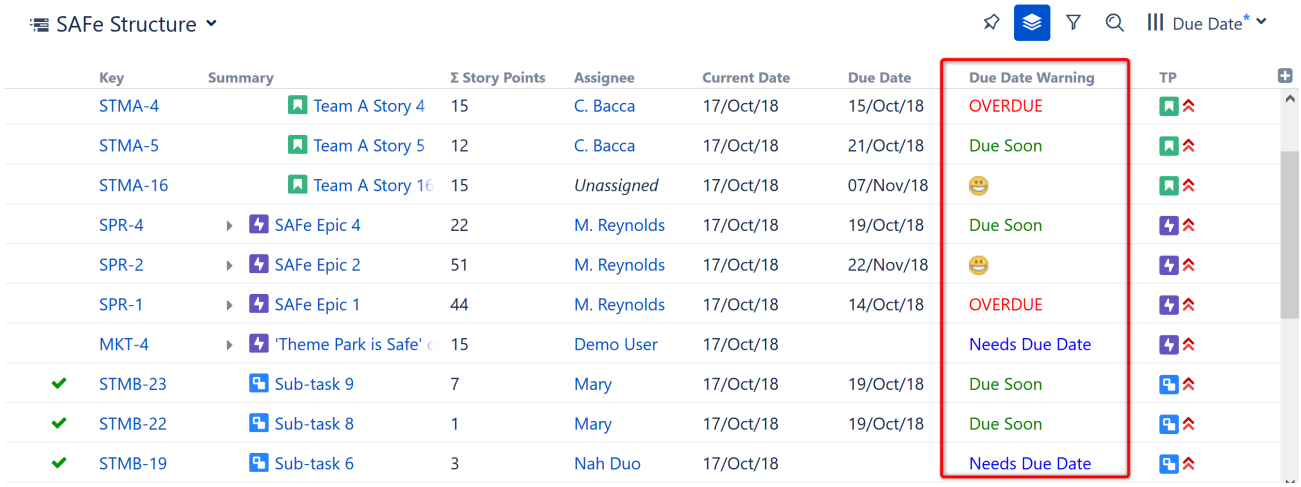
It's fine to mix text and emojis/images in the same column, just not the same cell.

Examples

Example 1: Progress Warnings

In the following example, we have created a simple formula to draw attention to overdue and upcoming due dates:

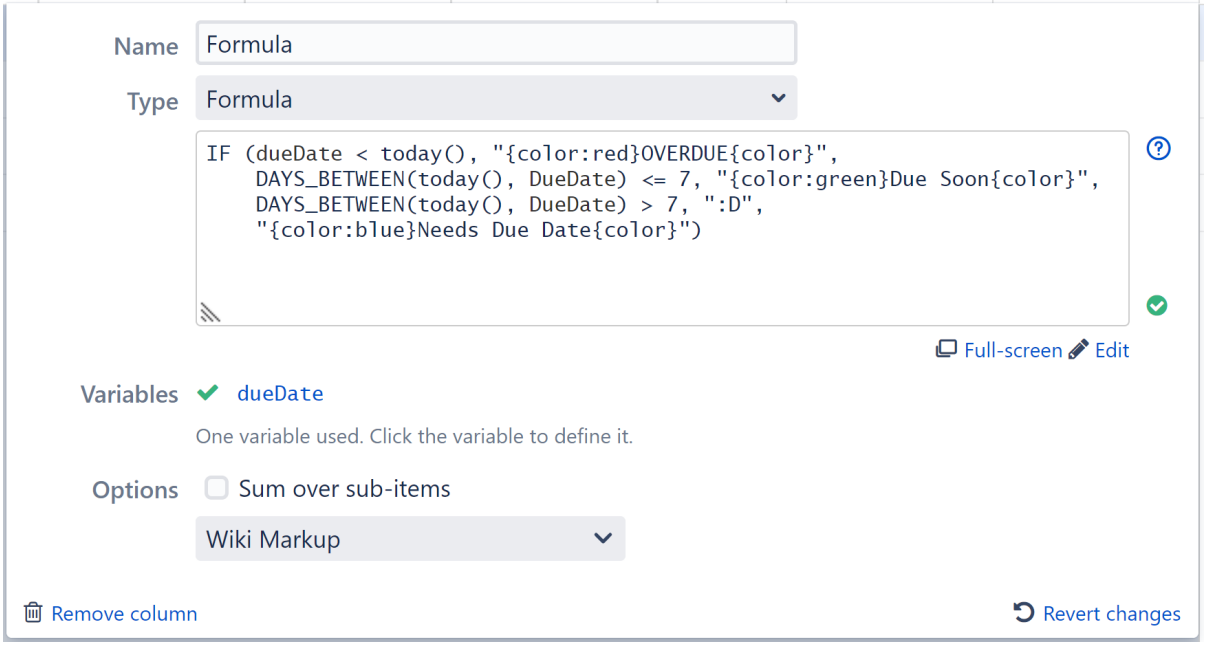
- When an issue is overdue, a red "OVERDUE" warning appears in the column
- When an issue is due within the next 7 days, the columns displays a green “Due Soon”
- When there's over a week to go, the issue gets a smiley face
- And if the issue doesn't have a due date, it let's you know that too



Key	Summary	Σ Story Points	Assignee	Current Date	Due Date	Due Date Warning	TP
STMA-4	Team A Story 4	15	C. Bacca	17/Oct/18	15/Oct/18	OVERDUE	
STMA-5	Team A Story 5	12	C. Bacca	17/Oct/18	21/Oct/18	Due Soon	
STMA-16	Team A Story 16	15	*Unassigned*	17/Oct/18	07/Nov/18	😃	
SPR-4	SAFe Epic 4	22	M. Reynolds	17/Oct/18	19/Oct/18	Due Soon	
SPR-2	SAFe Epic 2	51	M. Reynolds	17/Oct/18	22/Nov/18	😃	
SPR-1	SAFe Epic 1	44	M. Reynolds	17/Oct/18	14/Oct/18	OVERDUE	
MKT-4	'Theme Park is Safe'	15	Demo User	17/Oct/18		Needs Due Date	
STMB-23	Sub-task 9	7	Mary	17/Oct/18	19/Oct/18	Due Soon	
STMB-22	Sub-task 8	1	Mary	17/Oct/18	19/Oct/18	Due Soon	
STMB-19	Sub-task 6	3	Nah Duo	17/Oct/18		Needs Due Date	

To accomplish this, we added markup language to a standard **IF** statement:

```
IF (dueDate < today(), "{color:red}OVERDUE{color}",
    DAYS_BETWEEN(today(), DueDate) <= 7, "{color:green}Due Soon{color}",
    DAYS_BETWEEN(today(), DueDate) > 7, ":D",
    "{color:blue}Needs Due Date{color}")
```

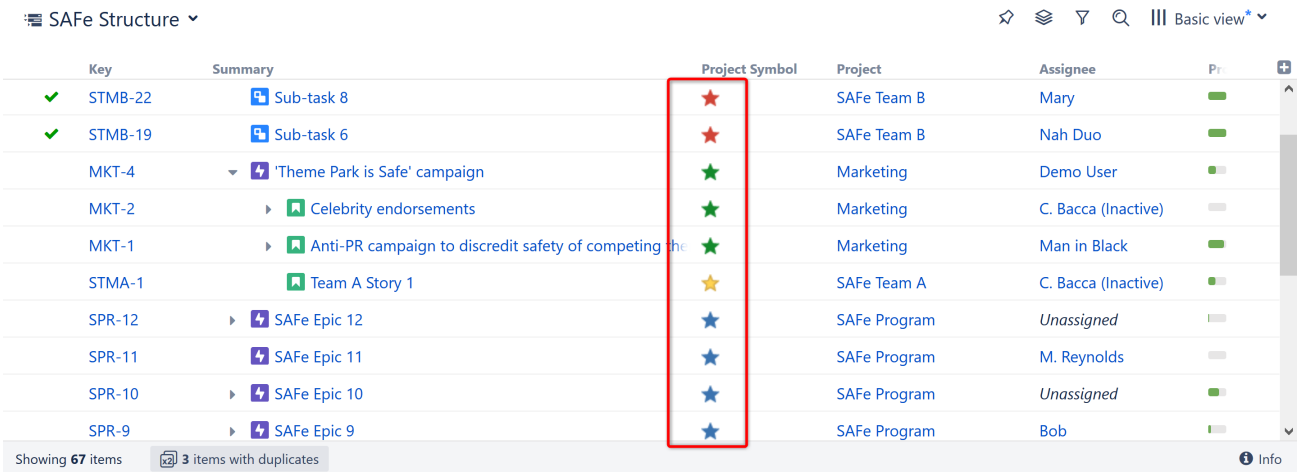


We used text to call attention to overdue items, but you could also add a flag: "(flag)"

To learn more about using If statements, DAYS_BETWEEN, or any other functions, see Expr Function Reference(see page 345).

Example 2: Project Markers

In this example, we've created a column to quickly identify each project we're working on. In this case, each project is marked by a unique star color.



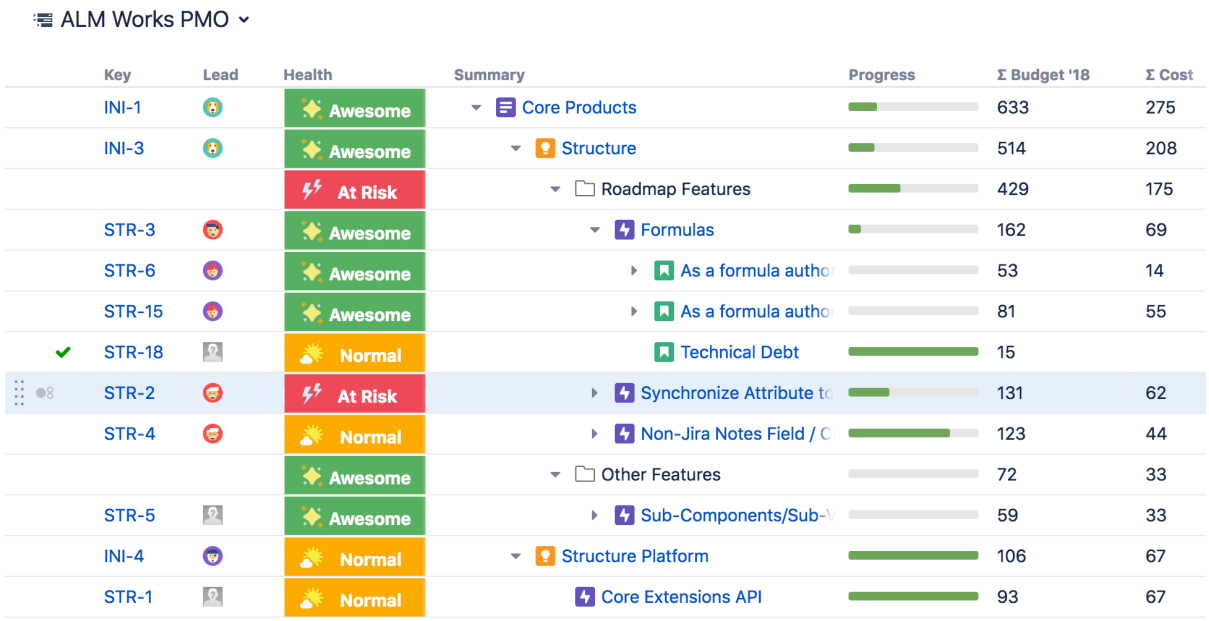
To create this column, we used the special character notations for stars "(*)" - along with color designations:

```
CASE(project, "SAFe Program", "(*b)", "SAFe Team A", "(*y)", "SAFe Team B", "(*r)",
"Marketing", "(*g)")
```

You could apply this same concept to any field, and you don't have to stick with stars. For example, you may want to color-code issues by team – or insert photos of your team mascots!

Example 3 - Issue Health

Here's the code for the issue health column you saw at the top of this article.

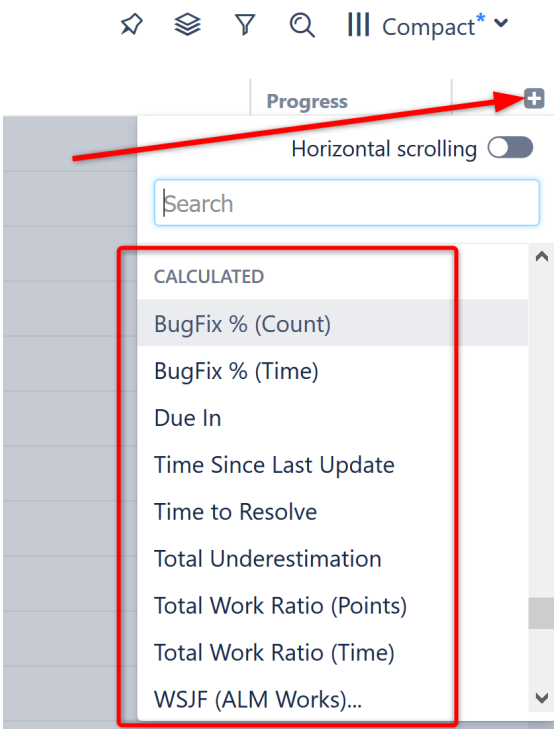


```
with WORK_TIME_TO_CALENDAR_TIME(time) = (
  with min = 60 * 1000:
  with hour = 60 * min:
  with day = 8 * hour:
  with week = 5 * day:
  with weeks = FLOOR(time / week):
  with r1 = time - weeks * week:
  with days = FLOOR(r1 / day):
  with r2 = r1 - days * day:
  with hours = FLOOR(r2 / hour):
  with r3 = r2 - hours * hour:
  with mins = r3 / min:
  (((weeks * 7 + days) * 24 + hours) * 60 + mins) * min
):

with FORMAT_CAPTION(image, color, caption, offset1, offset2) = (
  with SPACE(pixels) = """!https://upload.wikimedia.org/wikipedia/commons/5/52/
Spacer.gif|width=$pixels!""":
  """{panel:borderStyle=solid|borderColor=white|bgColor=$color} !$image|
width=20,height=20! ${SPACE(offset1)}{color:white}$caption{color}${SPACE(offset2)}
{panel}"""
):

with diff = FLOOR((due_date - (TODAY() + WORK_TIME_TO_CALENDAR_TIME(sum
{remaining}))) / 86400000):

IF type = "epic" or type = "epic enabler":
IF diff < 0:
  FORMAT_CAPTION("https://lh3.google.com/u/0/d/
19uzgf1XoSVOgJqXjoJKSjKGZggnfV83A=w2760-h1400-iv1", "#EF4B59", "*At Risk* ", 7, 7)
ELSE IF diff < 15:
  FORMAT_CAPTION("https://lh3.google.com/u/0/d/1pAa-
BvSvKCiPv1oplNFbkih1t7wuT2A6=w2880-h1430-iv1", "#FFAF00", " *OK* ", 18, 20)
ELSE:
  FORMAT_CAPTION("https://lh3.google.com/u/0/d/
1pNS0ctx-1T8z5s73rT_IsJmxV5fNhSKF=w2880-h1430-iv1", "#59B161", "*Great*", 13, 15)
```

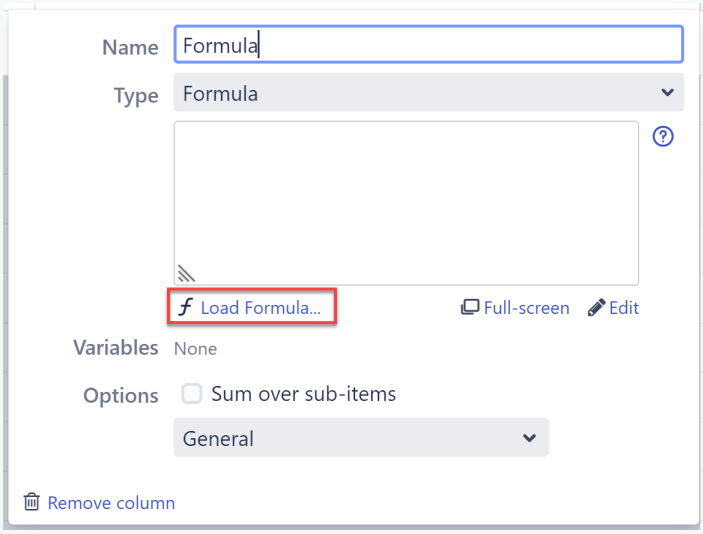
3.8.1.6 Bundled Formulas

The simplest way to include formulas in a structure is to use one of our bundled formulas.(see page 501)

To add a bundled formula, click the **+** button to the right of the column header and scroll down until you locate the **CALCULATED** section.



You can also add a bundled formula from the Formula column settings screen - simply select "Load Formula..."



The following predefined formulas are available:

Column Name	Description
Unresolved Blockers	Displays all issues that block the current issue and are unresolved.

Column Name	Description
BugFix % (Count)	Displays the percentage of bugs among all sub-issues. Bugs are identified by having issue type "Bug".
BugFix % (Time)	Displays the percentage of time scheduled and spent on bugs, compared to the time scheduled and spent on all sub-issues. Uses Jira time tracking fields.
Due In	Displays the amount of calendar time left before each issue's Due Date.
Issue Health	Provides a visual health status for each item, based on issue progress by time tracking.
Time Flagged	Displays the total time a task was flagged.
Time Since Last Update	Displays the amount of calendar time that has passed since the issue was last updated.
Tasks Since Sprint Started	Displays the number of tasks added since the last sprint began.
Time to Resolve	For resolved issues, displays the amount of calendar time that passed between issue creation and its resolution.
Total Underestimation	Displays the percentage by which the total actual time expenditure exceeded the total original estimate. Uses total Time Spent and Remaining Estimate fields to calculate the actual time. This "Totals" formula uses the SUM function(see page 387) to calculate a value for the issue and all its sub-issues.
Total Work Ratio (Points)	Ratio of total work done to the total amount of work. The amount of work is counted in Story Points, and issues are considered "done" when they have a non-empty Resolution field. This "Totals" formula uses the SUM function(see page 387) to calculate a value for the issue and all its sub-issues.
Total Work Ratio (Time)	Ratio of total work done to the total amount of work. The amount of work is based on the sum of Time Spent and Remaining Estimate values. This "Totals" formula uses the SUM function(see page 387) to calculate a value for the issue and all its sub-issues.
WSJF (Basic)	Weighted Shortest Job First metric, based on basic attributes available in any Jira – Priority, Votes, Watchers, Due Date, Story Points and Remaining Estimate.

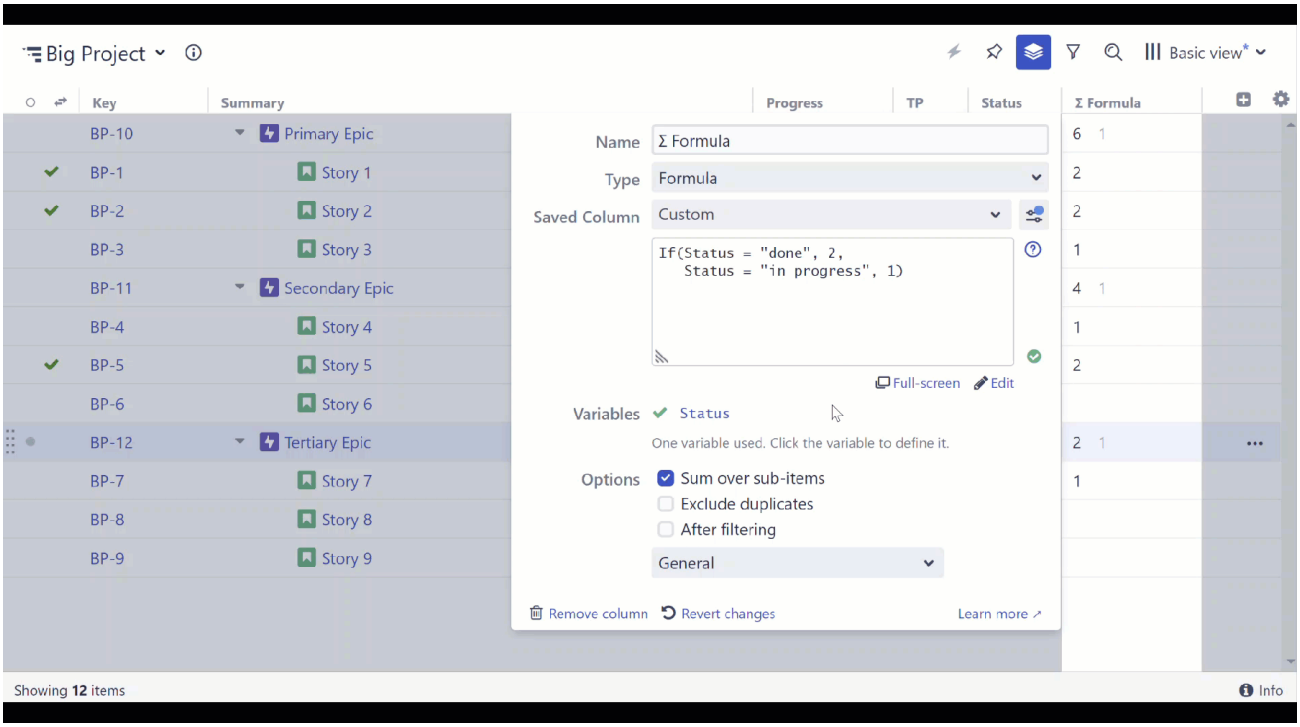
Column Name	Description
WSJF (SAFe)	Weighted Shortest Job First metric, based on recommendations from Scaled Agile Inc. To use this formula, you must set up the following numerical fields: • Job Size • User/Business Value • Time Criticality • Risk Reduction • Opportunity Enablement If you have such fields but they are not numeric (for example, a select list), edit the formula and replace the usage of a variable with a CASE() function, where you can assign individual numerical weights to each option.
WSJF (ALM Works)	Weighted Shortest Job First metric, according to categories used at ALM Works: • Benefit • Pain • Marketability • Impact • Cost • Risk • Clarity To use this formula, you must set up such fields with the following values: Nil, Low, Medium and High.
Σ Assignee Cost	Calculates the dollar amount of the task, based by the time (Time Spent + Remaining Estimate) multiplied by the per hour rate for the current Assignee. The rate is taken from the "Hourly Rate" additional property for the user who is the assignee. Shows the total amount for the issue and its sub-issues.

3.8.1.7 Saving a Formula

You can save formulas to reuse in another structure, share with others, or use in generators.

To save a formula:

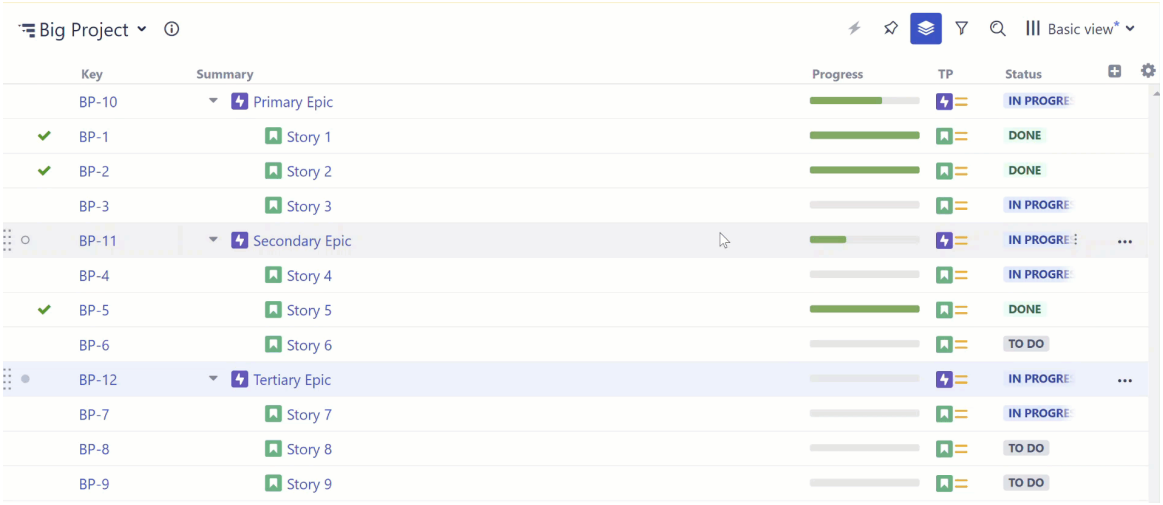
1. Complete your formula and make sure it works as expected.
2. Click the settings button beside the Saved Column selector.
3. Select **Save as...**
4. Enter a name and description for your formula column.
5. Choose whether the column should be private or global (available to anyone with Structure access).
6. Click **Save**.



3.8.1.8 Opening a Saved Formula

To open a saved formula:

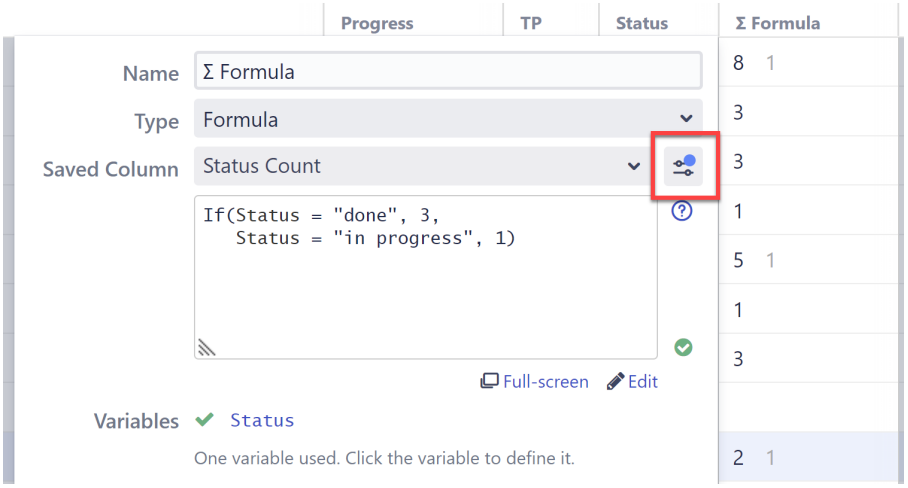
1. Add a new Formula column using the Add Column menu
2. Open the Saved Columns dropdown
3. Scroll through the list of saved columns, or use the search bar to locate the formula. *Note: if you don't find it, check the other lists (Bundled, Global, My Columns).*
4. Select your formula.



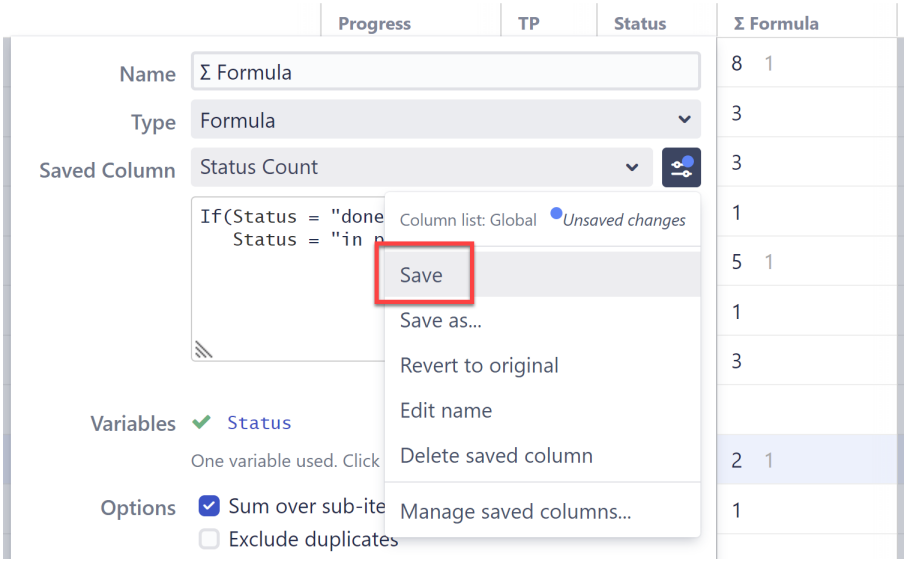
You can now use and even customize the formula. If you make changes to the formula, those won't be seen by anyone else using the formula unless you save them.

Saving Changes to a Saved Formula

If you've made changes to a saved formula, you'll see a blue indicator on the Saved Column settings button.



To update the saved file with those changes, click the settings button and select **Save**.

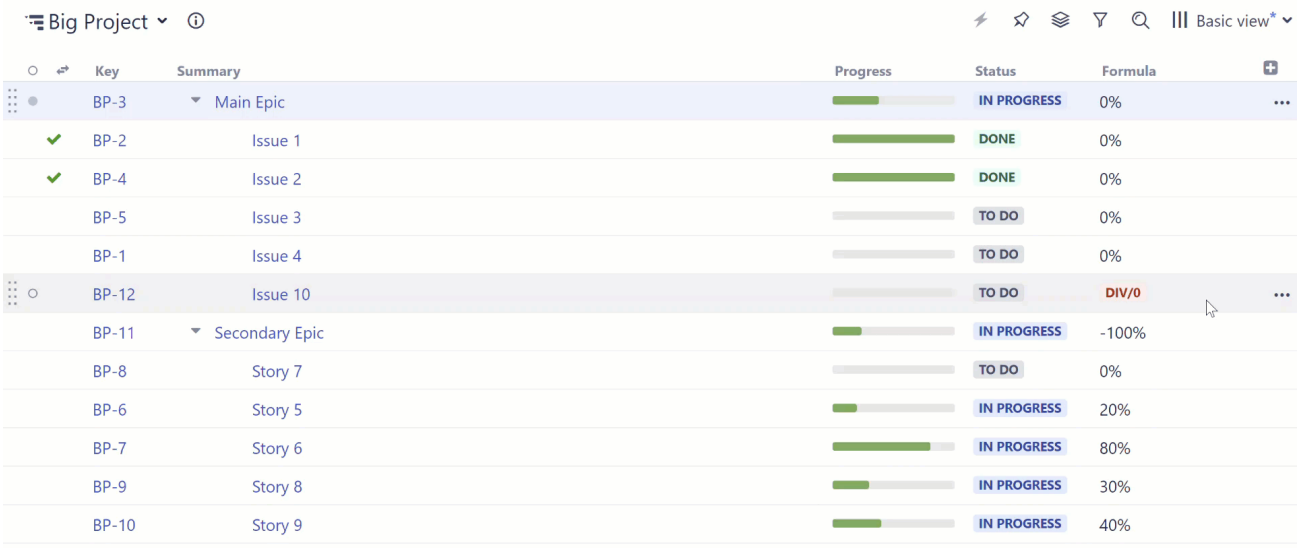


3.8.1.9 Debugging a Formula

If you are getting unexpected results from a formula, the debugging tool allows you to see the current value of each variable in a specific row.

To use the debugging tool:

1. Open the formula column configuration screen.
2. Click the debugging icon (it looks like a bug).
3. Select the row you want to inspect. This is most often a Jira issue, but could also be a folder(see page 104), memo(see page 105), or other item.
4. In the formula window, hover over a highlighted item to see its current value for the selected row.



In the example above, we can see that we're getting an error for "Issue 10" because there are no Original Estimate or Remaining Estimate values for that issue.

If you can't find the row you want to inspect in the dropdown, try the following:

1. Close the column configuration.
2. Scroll to the bottom of the structure (this forces all rows to load).
3. Reopen the column configuration and try again.

3.8.2 Sample Formulas

Analytics

3.8.2.1 Calculate time needed to burn down the backlog

Assuming your team can accomplish 10 story points a week, this will tell us how long (in weeks) it will take to burn down work and get to certain items further down the backlog. To adjust how many story points the team can work in a week, simply change the "velocity" value.

```
WITH velocity = 10:
  CONCAT(SUM#preceding{story_points} / velocity, "w")
```

For this example to work most effectively, the structure should be sorted based on how you choose which work to complete first. See Sort Generators(see page 263) for more information.

3.8.2.2 Calculate days past due

This example checks for items that are overdue and returns the number of days the item is overdue.

```
IF dueDate < NOW():
   DAYS_BETWEEN(dueDate, NOW()) CONCAT " days late"
```

3.8.2.3 Compare the original estimate to work logged and the remaining estimate

```
IF originalEstimate:
    (timeSpent + remainingEstimate) / originalEstimate
ELSE:
    "not estimated"
```

3.8.2.4 Calculate the interquartile range of story point estimates

```
WITH points = ARRAY { storyPoints } :   // Holds all the story points of the
children.
  QUARTILE(points, 3) - QUARTILE(points, 1)
```

Comments

3.8.2.5 Show the date, author, and text of the latest comment

```
comments.UMAX_BY($.created).map(CONCAT(
    $.author.user_display_name(),
    " said at ",
    FORMAT_DATETIME($.created, "yyyy-MM-dd HH:mm:ss"),
    ": ", $.body))
```

3.8.2.6 Show the last comment made by a user

```
comments.FILTER($.author = "admin").UMAX_BY($.created)
```

In this example, the last comment made by "admin" will be shown. To show comments for another user, replace "admin" accordingly.

3.8.2.7 Show the date of the latest comment done by a user

```
comments.filter(x -> x.author = "admin").map(x -> x.created).max()
```

In this example, the date corresponds to the last comment made by "admin." To show the date for another user, replace "admin" accordingly.

3.8.2.8 Display "Answered" if there are comments after my latest one

```
WITH myLastCommentDate = comments.FILTER($.author = me()).MAP($.created).MAX() :
   IF (comments.ANY($.created > myLastCommentDate); "Answered")
```

Historical Values

3.8.2.9 Show the historical value of an issue field at a specific date

In the example below, we're using the Due Date field. You can use any system or custom field.

```
historical_value(this, "duedate", datetime("15/May/18 6:24 PM"))
```

Note: this formula also uses the Datetime function - learn more(see page 263)

3.8.2.10 Show the number of tasks added since the last sprint began

```
SUM {
 IF history.changes
    .FILTER($.field = "sprint")
    .LAST()
    .changeGroup.timestamp > sprint.last().startDate: 1
}
```

3.8.2.11 Show who changed the field value

The example below shows who changed the Resolution field, but you can replace "resolution" with another system or custom field:

```
history.changes
    .FILTER($.field = “resolution”).last().changeGroup.author
```

3.8.2.12 Time Flagged: Time the task was marked with a flag

```
with flag_change_time(value) =
history.changes
  .filter($.field = "flagged")
  .filter($.to = value)
  .changeGroup.time :
with flag_on_time = flag_change_time("Impediment") :
with flag_off_time = flag_change_time("") :IF flag_on_time && flag_off_time :
flag_off_time - flag_on_time
ELSE IF flag_on_time : now() - flag_on_time
```

3.8.2.13 Time in status for a specific month

```
WITH year = 2023:
WITH month = 1: // 1 for Jan, 12 for Dec
WITH keyStatus = "in progress": // key-insensitive
WITH calendar = "Standard work calendar 8/5": // other option is Standard calendar
24/7, the value is locale-dependant, also Gantt calendars are available
WITH startDate = MAKE_DATE(year, month, 1):
WITH finishDate = MIN(DATE_ADD(startDate, 1, "month"), NOW()):
WITH isStart(change) = change.from != keyStatus AND change.to = keyStatus:
WITH isFinish(change) = change.from = keyStatus AND change.to != keyStatus:
WITH intervalFits(start, finish)
  =  start >= startDate AND start <= finishDate
  OR finish >= startDate AND finish <= finishDate
  OR start < startDate AND finish > finishDate:

WITH statusChanges = history.changes
  .FILTER($.field = "status" AND ($.isStart() OR $.isFinish())):
WITH times = MERGE_ARRAYS(
  IF statusChanges.FIRST().isFinish(): MIN(startDate,
statusChanges.FIRST().changeGroup.time),
  statusChanges.changeGroup.time,
  IF statusChanges.LAST().isStart(): MAX(finishDate,
statusChanges.LAST().changeGroup.time)
):

IF times: SEQUENCE(0, times.SIZE() - 1)
  .FILTER(MOD($, 2) == 0 AND intervalFits(times.GET($), times.GET($ + 1)))
  .MAP(CALENDAR_DURATION(MAX(times.GET($), startDate), MIN(times.GET($ + 1),
finishDate), calendar))
  .SUM()
```

Issue Links and Subtasks

3.8.2.14 Show linked issues

Displays issues linked to the current issue.

```
issueLinks.MAP(IF($.source = this, $.destination, $.source))
```

3.8.2.15 Show issue links

Displays issue links containing the current issue. Ex. STR-006 → GANTT-002

```
issueLinks.MAP($.source.key CONCAT '→' CONCAT $.destination.key)
```

3.8.2.16 Show issues blocking the current issue

Displays issue links for all blockers.

```
WITH _format(issue) = """[${issue.key}|${issue.url}]""" :
issuelinks
  .FILTER($.type = 'Blocks' AND $.destination = this)
  .MAP(_format($.source))
```

Make sure to set the column Options to Wiki Markup.

Want to display another link type? Change: `$.type = 'Blocks'`

3.8.2.17 Check whether all blocking issues are resolved

Displays "OK" if all issues linked via the "Blocks" link type are marked as resolved.

```
IF issueLinks.FILTER($.type = "Blocks" AND $.destination = this).ALL($.source.resolut
ion):
   "OK"
```

3.8.2.18 Show parent issue

Displays the parent issue of the current item, based on the "is parent of" link.

Depending on the direction of your parent links, select one of the following:

Outward parent links

```
issueLinks.FILTER($.type.outward = "is parent of" AND $.destination = this).MAP($.sou
rce.key CONCAT ' - ' CONCAT $.source.summary)
```

or

Inward parent links

```
issueLinks.FILTER($.type.inward = "is parent of" AND $.destination = this).MAP($.sour
ce.key CONCAT ' - ' CONCAT $.source.summary)
```

3.8.2.19 Show percent of subtasks that have been completed

```
IF subtasks.SIZE() > 0 :

    subtasks.FILTER($.status = 'Done').SIZE() / subtasks.SIZE()
```

Items and Properties

3.8.2.20 Access an item property

Use the following format: `item.property`

The following returns the release date for each fix version:

```
fixVersions.releaseDate
```

Note: if the fix version field contained multiple values, multiple dates will be returned.

For a list of accessible item types and their properties, see Item Property Reference(see page 406).

3.8.2.21 Get a custom field value for this issue, its epic, or its sub-task

You can accomplish this in a few different ways:

```
this.storypoints   // Using item properties. Use a lowercased custom field name, with
spaces skipped.
this.ACCESS("Story Points") // Using the ACCESS function. Write the custom field
exactly as it appears in Jira (with spaces).
this.customfield_###### // Using the custom field's id.
```

3.8.2.22 See how many sprints an issue has been added to

```
sprint.size()
```

3.8.2.23 Find the highest subtask priority

```
subtasks.priority.UMAX()
```

Returns the highest priority of the subtasks.

3.8.2.24 Find the subtask with the highest priority

```
with highest_priority = subtasks.priority.UMAX(): subtasks.FILTER($.priority =
highest_priority)
```

Returns all subtasks with the highest priority.

3.8.2.25 Compare two priorities

```
IF(priority1.sequence > priority2.sequence)
```

3.8.2.26 Predict the finish date for epics

```
IF issueType = epic :
  MAX(epicStories.sprint.endDate)
```

Returns the latest sprint end date for stories within each epic, even if those stories are not contained in the structure.

JQL and S-JQL

3.8.2.27 Show aggregate story points for a specific Jira user group

```
SUM {
  IF JQL { assignee in membersOf('Group A') } :
  storyPoints
}
```

Note: Replace 'Group A' with the name of the group you want to calculate for.

Want to aggregate another value? Just replace 'storyPoints' with the attribute you want to calculate.

Users

3.8.2.28 Show everyone who worked on the task

```
ARRAY(reporter, assignee, developer, tester)
```

Note: developer and tester are custom fields - they will be automatically mapped only if those custom fields exist in your Jira instance.

3.8.2.29 Show everyone who worked on any task in the subtree

```
VALUES { ARRAY(reporter, assignee, developer, tester) }
```

Note: developer and tester are custom fields - they will be automatically mapped only if those custom fields exist in your Jira instance.

3.8.2.30 Calculate who logged the most work

```
worklogs
  .GROUP($.author)
  .MAP(ARRAY($.group, $.elements.timespent.sum()))
  .UMAX_BY($.GET(1))
  .GET(0)
```

3.8.2.31 Get a detailed description of the tasks users spent time on

First, use an attribute grouper(see page 175) with the formula:

```
worklogs.author.UNIQUE()
```

Then, add a formula column:

```
IF itemType = 'user':
SUM#children {
  WITH user = PARENT { item } :
  worklogs
    .FILTER($.author = user)
    .timeSpent.SUM()
}
```

Versions

3.8.2.32 Check for a specific fix version

```
fixVersions.CONTAINS("v1")
```

If the issue contains that fixVersion, returns 1 (true). Otherwise, returns 0 (false).

3.8.2.33 Get the latest/earliest fix version

```
fixVersions.UMAX_BY($.releaseDate) // latest


fixVersions.UMIN_BY($.releaseDate) // earliest
```

3.8.2.34 Find the largest time span of an affected version

```
affectedVersions.MAP(IF $.releaseDate AND $.startDate: $.releaseDate -
$.startDate).MAX()
```

For each Affected Version, subtracts the Start Date from the Release Date, and returns the Affected Version with the largest result.

Want the shortest result? Change MAX to MIN.

Learn more: MAP function(see page 263)

3.8.2.35 Show all versions referenced in the subtree

```
VALUES { ARRAY(fixVersions, affectedVersions).FLATTEN().UNIQUE() }
```

3.8.2.36 Get all fix versions with future release dates

```
fixVersions.FILTER($.releaseDate AND $.releaseDate > NOW())
```

3.8.2.37 Show all released affected versions

```
affectedVersions.FILTER($.isreleased)
```

3.8.2.38 Show all issues released during a set period of time

When used as a filter generator or transformation, the following code will show only issues that were part of fix versions released during Q1, 2021.

```
DATE(“0/Jan/2021”) < fixVersion.releaseDate

   AND fixVersion.releaseDate < DATE (“31/Mar/2021”)
```

3.8.2.39 Check that child issues and paret issues have the same Fixversion

```
with parentVersion = PARENT{FixVersion}:
  if(parentVersion and !parentVersion.contains(fixVersion); "version mismatch")
```

Wiki Markup

Wiki markup(see page 251) allows you to get creative and visualize more complex metrics in Structure columns, such as custom progress bars, bar charts and much more.

We've put together several advanced, customizable examples of wiki markup usage:

3.8.2.40 Display subtasks as links

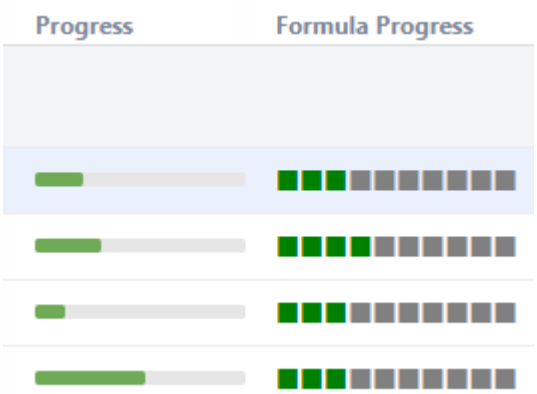
```
subtasks.MAP("""[${$.key}|${$.url}]""")
```

3.8.2.41 Create a borderless background behind the value

```
WITH addBackground(value, color) =
  """{panel:bgColor=$color|borderWidth=0px}$value{panel}""":
addBackground(summary, "#ADFF2F")
```

3.8.2.42 Customizable Progress Bar

In this simple example, we used Wiki Markup to create a customized progress bar. In the left column you can see the built-in progress column. In the right one, we've built a progress bar which is split into 10% sections.



We used the following formula to build the custom progress bar:

Simple progress bar

```
WITH simpleProgressBar(progress, maxProgress, stepCount) = (
  WITH _bars(count, color) = """{color:$color}${REPEAT("■", count)}{color}""":
  WITH doneBarsCount = FLOOR(progress / maxProgress * stepCount):
  _bars(doneBarsCount, "green") CONCAT _bars(stepCount - doneBarsCount, "gray")
):

simpleProgressBar(customProgress, 1, 10)
```

Starting with this, you can tailor the progress bar to your team's particular needs.

- Colors can easily be configured by altering the "color" values - in this case, we used green and gray squares.
- The progress calculation can be based on any percentage value. In the following example, we used an arbitrary percentage field and aggregated up the hierarchy.

Simple progress bar

```
WITH simpleProgressBar(progress, maxProgress, stepCount) = (
  WITH _bars(count, color) = """{color:$color}${REPEAT("■", count)}{color}""":
  WITH doneBarsCount = FLOOR(progress / maxProgress * stepCount):
  _bars(doneBarsCount, "green") CONCAT _bars(stepCount - doneBarsCount, "gray")
):

simpleProgressBar(SUM { progressField }, SUM { 1 }, 10)
```

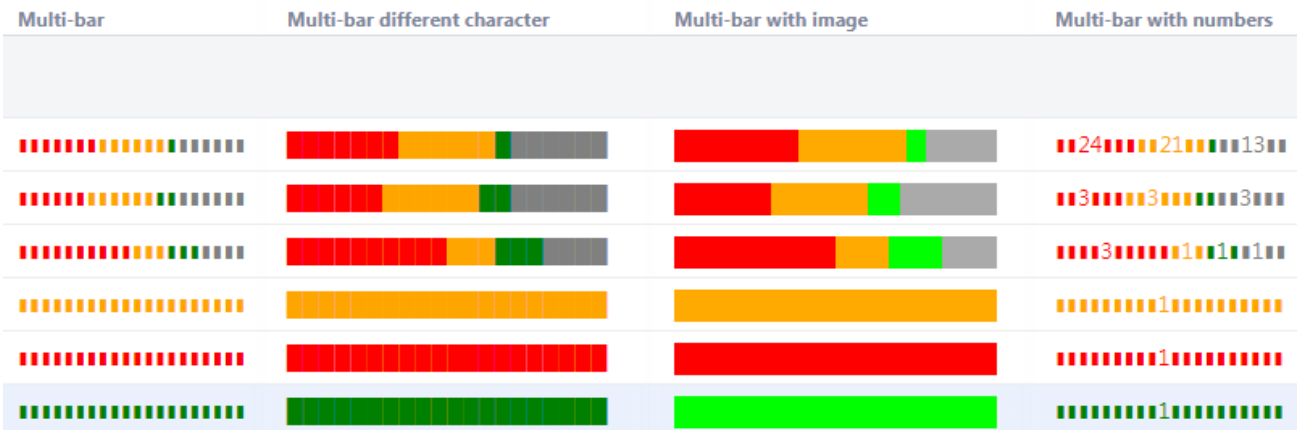
This can be especially useful if you want to display progress based on some complex fields, like a ScriptRunner scripted field, which is not supported by the standard formula column at the moment.

3.8.2.43 Customizable Status Bars

Wiki markup can also be used to create more complex progress calculations, based on multiple issue statuses.

In the following example, we created multiple custom status bars, tracking the following statuses:

- To Do = Red
- In Progress = Orange
- Done = Green
- All Other Statuses = Gray



As with our custom progress bar, these formulas can easily be modified to adjust status colors, include additional statuses or represent each status in a different format.

3.8.2.44 Multi-bar

We used the following code to build the Multi-bar Status Bar.

Multi-tiered progress bar

```
//stepCount - length of the bar chart in characters
WITH multiProgressBar(progressArray, maxProgress, colorsArray, colorForRemaining,
stepCount) = (
  WITH _bars(count, color) = (IF count > 0: """{color:$color}${REPEAT("▌", count)}
{color}""" ELSE ""):
  WITH barCounts = progressArray.MAP(FLOOR($ / maxProgress * stepCount)):
  progressArray.INDEXES()
    .MAP(_bars(barCounts.GET($), colorsArray.GET($)))
    .MERGE_ARRAYS(_bars(MAX(0, stepCount - barCounts.SUM()), colorForRemaining))
    .JOIN("", "", "")
):

WITH todo = COUNT#truthy { status = "To Do" }:
WITH inProgress = COUNT#truthy { status = "In Progress" }:
WITH done = COUNT#truthy { status = "Done" }:

multiProgressBar(
  ARRAY(todo, inProgress, done), COUNT { 1 },
  ARRAY("red", "orange", "green"), "gray",
  20
)
```

You can change the appearance of the status simply by altering the granularity (length of the bar sections) or a using a larger symbol as we did in the **Multi-bar different character** example.

While the ■ or ▌ symbols may lack solid feel, the ▉ symbol still creates a slight brick-layer effect.

3.8.2.45 Multi-bar with Image

In this example, we used a simple, monochrome images (a 1x1 pixel size is enough) to make the status bar appear more solid. If you decide to try this, we highly recommend using a locally-hosted image, rather than one taken from public sources, because some hosts may block multiple successive requests for an image.

Multi-tiered progress bar based on images

```
//Granularity - length of the bar chart in pixels
WITH multiProgressBarWithImage(progressArray, maxProgress, imagesArray,
imageForRemaining, granularity) = (
  WITH bar(width, image) = (IF width > 0: """!$image|height=20,width=$width!""" ELSE
""):
  WITH barCounts = progressArray.MAP(FLOOR($ / maxProgress * granularity)):
  progressArray.INDEXES()
    .MAP(bar(barCounts.GET($), imagesArray.GET($)))
    .MERGE_ARRAYS(bar(MAX(0, granularity - barCounts.SUM()), imageForRemaining))
    .JOIN("", "", "")
):


WITH todo = COUNT#truthy {status = "to do"}:
WITH inProgress = COUNT#truthy {status = "in progress"}:
WITH done = COUNT#truthy {status = "done"}:

WITH link(name) = """https://www.example.com/images/$name.png""":

multiProgressBarWithImage(
  ARRAY(todo, inProgress, done), COUNT{1},
  ARRAY("Red", "Orange", "Green").MAP(link), link("Gray"),
  200
)
```

3.8.2.46 Multi-bar with Numbers

In this last example, the status bar displays an issue count for each status, when the bar width permits. This code could be easily customized to display either the actual number of issues or their percentage.

Progress bar with numbers

```
//Parameters: granularity - length of bar-chart in characters; bar - filler of the
bar chart
WITH multiProgressBarWithNumbers(progressArray, maxProgress, colorsArray,
colorForRemaining, granularity, bar) = (
  WITH bars(count, value, color) = (
    IF count <= 0:
      ""
    ELSE:
      WITH bars = (
        WITH charsForValue = LEN(value):
        IF count >= charsForValue + 2:
          WITH charsBeforeValue = FLOOR((count - charsForValue) / 2):
          REPEAT(bar, charsBeforeValue)
          CONCAT value
          CONCAT REPEAT(bar, count - charsBeforeValue - charsForValue)
        ELSE:
          REPEAT(bar, count)
      ):
      """{color:$color}$bars{color}"""
  ):
  WITH barCounts = progressArray.MAP(FLOOR($ / maxProgress * granularity)):
  progressArray.INDEXES()
    .MAP(bars(barCounts.GET($), progressArray.GET($), colorsArray.GET($)))
    .MERGE_ARRAYS(bars(MAX(0, granularity - barCounts.SUM()), MAX(0, maxProgress -
progressArray.SUM()), colorForRemaining))
    .JOIN("", "", "")
):

WITH todo = COUNT#truthy {status = "to do"}:
WITH inProgress = COUNT#truthy {status = "in progress"}:
WITH done = COUNT#truthy {status = "done"}:

multiProgressBarWithNumbers(
  ARRAY(todo, inProgress, done), COUNT{1},
  ARRAY("red", "orange", "green"), "gray",
  20, "▌"
)
```

3.8.2.47 Simple Burn-down Chart

You can get even more creative and use wiki markup to build mini-charts – including this simple burn-down chart. In this example, our chart displays created issues in red and resolved issues in green, with each pair corresponding to one day in a week.

Due to space limitations, there is a height limit of 20 pixels imposed within the chart, but this is more than enough to create a simple, powerful visualization.

Burn-down chart

```
WITH dataArray = ARRAY(
  COUNT#truthy {DATE_SUBTRACT(NOW(),6,"days") <= created  and DATE_SUBTRACT(NOW(),5,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),6,"days") <= resolved and DATE_SUBTRACT(NOW(),5,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),5,"days") <= created  and DATE_SUBTRACT(NOW(),4,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),5,"days") <= resolved and DATE_SUBTRACT(NOW(),4,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),4,"days") <= created  and DATE_SUBTRACT(NOW(),3,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),4,"days") <= resolved and DATE_SUBTRACT(NOW(),3,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),3,"days") <= created  and DATE_SUBTRACT(NOW(),2,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),3,"days") <= resolved and DATE_SUBTRACT(NOW(),2,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),2,"days") <= created  and DATE_SUBTRACT(NOW(),1,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),2,"days") <= resolved and DATE_SUBTRACT(NOW(),1,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),1,"days") <= created  and DATE_SUBTRACT(NOW(),8,"
hours") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),1,"days") <= resolved and DATE_SUBTRACT(NOW(),8,"
hours") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),8,"hours") <= created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),8,"hours") <= resolved}
):


//25 is maximum working height
WITH maxHeight = 25:
WITH maxValue = dataArray.MAX():
WITH getPicture(index) = (IF index.MOD(2) == 0: "https://www.example.com/images/
Red.png" ELSE: "https://www.example.com/images/Green.png"):
WITH getHeight(index) = IF maxValue : FLOOR(dataArray.GET(index) / maxValue *
maxHeight) ELSE : 0 :

IF itemtype != "issue":
dataArray
  .INDEXES()
  .MAP("""!${getPicture($)}|height=${getHeight($)},width=5!""")
  .JOIN("", "", "")
```

The criteria for issue inclusion can be easily customized to your team's needs. As mentioned above, we recommend hosting image files locally.

3.8.2.48 Sample Formulas - Analytics

Calculate time needed to burn down the backlog

Assuming your team can accomplish 10 story points a week, this will tell us how long (in weeks) it will take to burn down work and get to certain items further down the backlog. To adjust how many story points the team can work in a week, simply change the "velocity" value.

```
WITH velocity = 10:
  CONCAT(SUM#preceding{story_points} / velocity, "w")
```

For this example to work most effectively, the structure should be sorted based on how you choose which work to complete first. See Sort Generators(see page 280) for more information.

Calculate days past due

This example checks for items that are overdue and returns the number of days the item is overdue.

```
IF dueDate < NOW():
   DAYS_BETWEEN(dueDate, NOW()) CONCAT " days late"
```

Compare the original estimate to work logged and the remaining estimate

```
IF originalEstimate:
    (timeSpent + remainingEstimate) / originalEstimate
ELSE:
    "not estimated"
```

Calculate the interquartile range of story point estimates

```
WITH points = ARRAY { storyPoints } :   // Holds all the story points of the
children.
  QUARTILE(points, 3) - QUARTILE(points, 1)
```

3.8.2.49 Sample Formulas - Comments

Show the date, author, and text of the latest comment

```
comments.UMAX_BY($.created).map(CONCAT(
    $.author.user_display_name(),
    " said at ",
    FORMAT_DATETIME($.created, "yyyy-MM-dd HH:mm:ss"),
    ": ", $.body))
```

Show the last comment made by a user

```
comments.FILTER($.author = "admin").UMAX_BY($.created)
```

In this example, the last comment made by "admin" will be shown. To show comments for another user, replace "admin" accordingly.

Show the date of the latest comment done by a user

```
comments.filter(x -> x.author = "admin").map(x -> x.created).max()
```

In this example, the date corresponds to the last comment made by "admin." To show the date for another user, replace "admin" accordingly.

Display "Answered" if there are comments after my latest one

```
WITH myLastCommentDate = comments.FILTER($.author = me()).MAP($.created).MAX() :
   IF (comments.ANY($.created > myLastCommentDate); "Answered")
```

3.8.2.50 Sample Formulas - Historical Values

Show the historical value of an issue field at a specific date

In the example below, we're using the Due Date field. You can use any system or custom field.

```
historical_value(this, "duedate", datetime("15/May/18 6:24 PM"))
```

Note: this formula also uses the Datetime function - learn more(see page 281)

Show the number of tasks added since the last sprint began

```
SUM {
 IF history.changes
    .FILTER($.field = "sprint")
    .LAST()
    .changeGroup.timestamp > sprint.last().startDate: 1
}
```

Show who changed the field value

The example below shows who changed the Resolution field, but you can replace "resolution" with another system or custom field:

```
history.changes
    .FILTER($.field = “resolution”).last().changeGroup.author
```

Time Flagged: Time the task was marked with a flag

```
with flag_change_time(value) =
history.changes
  .filter($.field = "flagged")
  .filter($.to = value)
  .changeGroup.time :
with flag_on_time = flag_change_time("Impediment") :
with flag_off_time = flag_change_time("") :IF flag_on_time && flag_off_time :
flag_off_time - flag_on_time
ELSE IF flag_on_time : now() - flag_on_time
```

Time in status for a specific month

```
WITH year = 2023:
WITH month = 1: // 1 for Jan, 12 for Dec
WITH keyStatus = "in progress": // key-insensitive
WITH calendar = "Standard work calendar 8/5": // other option is Standard calendar
24/7, the value is locale-dependant, also Gantt calendars are available
WITH startDate = MAKE_DATE(year, month, 1):
WITH finishDate = MIN(DATE_ADD(startDate, 1, "month"), NOW()):
WITH isStart(change) = change.from != keyStatus AND change.to = keyStatus:
WITH isFinish(change) = change.from = keyStatus AND change.to != keyStatus:
WITH intervalFits(start, finish)
  =  start >= startDate AND start <= finishDate
  OR finish >= startDate AND finish <= finishDate
  OR start < startDate AND finish > finishDate:

WITH statusChanges = history.changes
  .FILTER($.field = "status" AND ($.isStart() OR $.isFinish())):
WITH times = MERGE_ARRAYS(
  IF statusChanges.FIRST().isFinish(): MIN(startDate,
statusChanges.FIRST().changeGroup.time),
  statusChanges.changeGroup.time,
  IF statusChanges.LAST().isStart(): MAX(finishDate,
statusChanges.LAST().changeGroup.time)
):

IF times: SEQUENCE(0, times.SIZE() - 1)
  .FILTER(MOD($, 2) == 0 AND intervalFits(times.GET($), times.GET($ + 1)))
  .MAP(CALENDAR_DURATION(MAX(times.GET($), startDate), MIN(times.GET($ + 1),
finishDate), calendar))
  .SUM()
```

3.8.2.51 Sample Formulas - Items and Properties

Access an item property

Use the following format: `item.property`

The following returns the release date for each fix version:

```
fixVersions.releaseDate
```

Note: if the fix version field contained multiple values, multiple dates will be returned.

For a list of accessible item types and their properties, see Item Property Reference(see page 406).

Get a custom field value for this issue, its epic, or its sub-task

You can accomplish this in a few different ways:

```
this.storypoints   // Using item properties. Use a lowercased custom field name, with
spaces skipped.
this.ACCESS("Story Points") // Using the ACCESS function. Write the custom field
exactly as it appears in Jira (with spaces).
this.customfield_###### // Using the custom field's id.
```

See how many sprints an issue has been added to

```
sprint.size()
```

Find the highest subtask priority

```
subtasks.priority.UMAX()
```

Returns the highest priority of the subtasks.

Find the subtask with the highest priority

```
with highest_priority = subtasks.priority.UMAX(): subtasks.FILTER($.priority =
highest_priority)
```

Returns all subtasks with the highest priority.

Compare two priorities

```
IF(priority1.sequence > priority2.sequence)
```

Predict the finish date for epics

```
IF issueType = epic :
  MAX(epicStories.sprint.endDate)
```

Returns the latest sprint end date for stories within each epic, even if those stories are not contained in the structure.

3.8.2.52 Sample Formulas - Issue Links and Subtasks

Show linked issues

Displays issues linked to the current issue.

```
issueLinks.MAP(IF($.source = this, $.destination, $.source))
```

Show issue links

Displays issue links containing the current issue. Ex. STR-006 → GANTT-002

```
issueLinks.MAP($.source.key CONCAT '→' CONCAT $.destination.key)
```

Show issues blocking the current issue

Displays issue links for all blockers.

```
WITH _format(issue) = """[${issue.key}|${issue.url}]""" :
issuelinks
  .FILTER($.type = 'Blocks' AND $.destination = this)
  .MAP(_format($.source))
```

Make sure to set the column Options to Wiki Markup.

Want to display another link type? Change: `$.type = 'Blocks'`

Check whether all blocking issues are resolved

Displays "OK" if all issues linked via the "Blocks" link type are marked as resolved.

```
IF issueLinks.FILTER($.type = "Blocks" AND $.destination = this).ALL($.source.resolut
ion):
   "OK"
```

Show parent issue

Displays the parent issue of the current item, based on the "is parent of" link.

Depending on the direction of your parent links, select one of the following:

Outward parent links

```
issueLinks.FILTER($.type.outward = "is parent of" AND $.destination = this).MAP($.sou
rce.key CONCAT ' - ' CONCAT $.source.summary)
```

or

Inward parent links

```
issueLinks.FILTER($.type.inward = "is parent of" AND $.destination = this).MAP($.sour
ce.key CONCAT ' - ' CONCAT $.source.summary)
```

Show percent of subtasks that have been completed

```
IF subtasks.SIZE() > 0 :

    subtasks.FILTER($.status = ‘Done’).SIZE() / subtasks.SIZE()
```

3.8.2.53 Sample Formulas - JQL and S-JQL

Show aggregate story points for a specific Jira user group

```
SUM {
  IF JQL { assignee in membersOf('Group A') } :
  storyPoints
}
```

Note: Replace 'Group A' with the name of the group you want to calculate for.

Want to aggregate another value? Just replace 'storyPoints' with the attribute you want to calculate.

3.8.2.54 Sample Formulas - Users

Show everyone who worked on the task

```
ARRAY(reporter, assignee, developer, tester)
```

Note: developer and tester are custom fields - they will be automatically mapped only if those custom fields exist in your Jira instance.

Show everyone who worked on any task in the subtree

```
VALUES { ARRAY(reporter, assignee, developer, tester) }
```

Note: developer and tester are custom fields - they will be automatically mapped only if those custom fields exist in your Jira instance.

Calculate who logged the most work

```
worklogs
  .GROUP($.author)
  .MAP(ARRAY($.group, $.elements.timespent.sum()))
  .UMAX_BY($.GET(1))
  .GET(0)
```

Get a detailed description of the tasks users spent time on

First, use an attribute grouper(see page 175) with the formula:

```
worklogs.author.UNIQUE()
```

Then, add a formula column:

```
IF itemType = 'user':
SUM#children {
  WITH user = PARENT { item } :
  worklogs
    .FILTER($.author = user)
    .timeSpent.SUM()
}
```

3.8.2.55 Sample Formulas - Versions

Check for a specific fix version

```
fixVersions.CONTAINS("v1")
```

If the issue contains that fixVersion, returns 1 (true). Otherwise, returns 0 (false).

Get the latest/earliest fix version

```
fixVersions.UMAX_BY($.releaseDate) // latest


fixVersions.UMIN_BY($.releaseDate) // earliest
```

Find the largest time span of an affected version

```
affectedVersions.MAP(IF $.releaseDate AND $.startDate: $.releaseDate -
$.startDate).MAX()
```

For each Affected Version, subtracts the Start Date from the Release Date, and returns the Affected Version with the largest result.

Want the shortest result? Change MAX to MIN.

Learn more: MAP function(see page 287)

Show all versions referenced in the subtree

```
VALUES { ARRAY(fixVersions, affectedVersions).FLATTEN().UNIQUE() }
```

Get all fix versions with future release dates

```
fixVersions.FILTER($.releaseDate AND $.releaseDate > NOW())
```

Show all released affected versions

```
affectedVersions.FILTER($.isreleased)
```

Show all issues released during a set period of time

When used as a filter generator or transformation, the following code will show only issues that were part of fix versions released during Q1, 2021.

```
DATE(“0/Jan/2021”) < fixVersion.releaseDate

   AND fixVersion.releaseDate < DATE (“31/Mar/2021”)
```

Check that child issues and paret issues have the same Fixversion

```
with parentVersion = PARENT{FixVersion}:
  if(parentVersion and !parentVersion.contains(fixVersion); "version mismatch")
```

3.8.2.56 Sample Formulas - Wiki Markup

Wiki markup(see page 251) allows you to get creative and visualize more complex metrics in Structure columns, such as custom progress bars, bar charts and much more.

We've put together several advanced, customizable examples of wiki markup usage:

Display subtasks as links

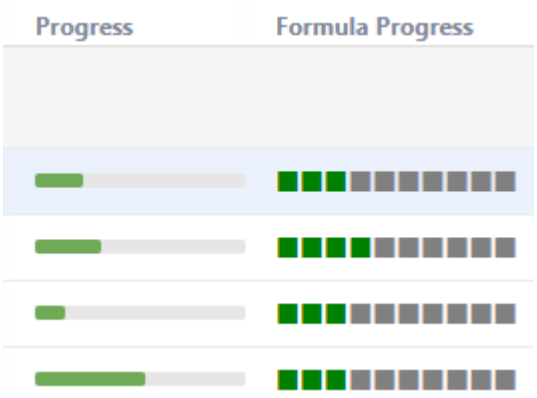
```
subtasks.MAP("""[${$.key}|${$.url}]""")
```

Create a borderless background behind the value

```
WITH addBackground(value, color) =
  """{panel:bgColor=$color|borderWidth=0px}$value{panel}""":
addBackground(summary, "#ADFF2F")
```

Customizable Progress Bar

In this simple example, we used Wiki Markup to create a customized progress bar. In the left column you can see the built-in progress column. In the right one, we've built a progress bar which is split into 10% sections.



We used the following formula to build the custom progress bar:

Simple progress bar

```
WITH simpleProgressBar(progress, maxProgress, stepCount) = (
  WITH _bars(count, color) = """{color:$color}${REPEAT("■", count)}{color}""":
  WITH doneBarsCount = FLOOR(progress / maxProgress * stepCount):
  _bars(doneBarsCount, "green") CONCAT _bars(stepCount - doneBarsCount, "gray")
):

simpleProgressBar(customProgress, 1, 10)
```

Starting with this, you can tailor the progress bar to your team's particular needs.

- Colors can easily be configured by altering the "color" values - in this case, we used green and gray squares.
- The progress calculation can be based on any percentage value. In the following example, we used an arbitrary percentage field and aggregated up the hierarchy.

Simple progress bar

```
WITH simpleProgressBar(progress, maxProgress, stepCount) = (
  WITH _bars(count, color) = """{color:$color}${REPEAT("■", count)}{color}""":
  WITH doneBarsCount = FLOOR(progress / maxProgress * stepCount):
  _bars(doneBarsCount, "green") CONCAT _bars(stepCount - doneBarsCount, "gray")
):

simpleProgressBar(SUM { progressField }, SUM { 1 }, 10)
```

This can be especially useful if you want to display progress based on some complex fields, like a ScriptRunner scripted field, which is not supported by the standard formula column at the moment.

Customizable Status Bars

Wiki markup can also be used to create more complex progress calculations, based on multiple issue statuses.

In the following example, we created multiple custom status bars, tracking the following statuses:

- To Do = Red
- In Progress = Orange
- Done = Green
- All Other Statuses = Gray

As with our custom progress bar, these formulas can easily be modified to adjust status colors, include additional statuses or represent each status in a different format.

Multi-bar

We used the following code to build the Multi-bar Status Bar.

Multi-tiered progress bar

```
//stepCount - length of the bar chart in characters
WITH multiProgressBar(progressArray, maxProgress, colorsArray, colorForRemaining,
stepCount) = (
  WITH _bars(count, color) = (IF count > 0: """{color:$color}${REPEAT("▌", count)}
{color}""" ELSE ""):
  WITH barCounts = progressArray.MAP(FLOOR($ / maxProgress * stepCount)):
  progressArray.INDEXES()
    .MAP(_bars(barCounts.GET($), colorsArray.GET($)))
    .MERGE_ARRAYS(_bars(MAX(0, stepCount - barCounts.SUM()), colorForRemaining))
    .JOIN("", "", "")
):

WITH todo = COUNT#truthy { status = "To Do" }:
WITH inProgress = COUNT#truthy { status = "In Progress" }:
WITH done = COUNT#truthy { status = "Done" }:

multiProgressBar(
  ARRAY(todo, inProgress, done), COUNT { 1 },
  ARRAY("red", "orange", "green"), "gray",
  20
)
```

You can change the appearance of the status simply by altering the granularity (length of the bar sections) or a using a larger symbol as we did in the **Multi-bar different character** example.

While the ■ or ▌ symbols may lack solid feel, the ▉ symbol still creates a slight brick-layer effect.

Multi-bar with Image

In this example, we used a simple, monochrome images (a 1x1 pixel size is enough) to make the status bar appear more solid. If you decide to try this, we highly recommend using a locally-hosted image, rather than one taken from public sources, because some hosts may block multiple successive requests for an image.

Multi-tiered progress bar based on images

```
//Granularity - length of the bar chart in pixels
WITH multiProgressBarWithImage(progressArray, maxProgress, imagesArray,
imageForRemaining, granularity) = (
  WITH bar(width, image) = (IF width > 0: """!$image|height=20,width=$width!""" ELSE
""):
  WITH barCounts = progressArray.MAP(FLOOR($ / maxProgress * granularity)):
  progressArray.INDEXES()
    .MAP(bar(barCounts.GET($), imagesArray.GET($)))
    .MERGE_ARRAYS(bar(MAX(0, granularity - barCounts.SUM()), imageForRemaining))
    .JOIN("", "", "")
):


WITH todo = COUNT#truthy {status = "to do"}:
WITH inProgress = COUNT#truthy {status = "in progress"}:
WITH done = COUNT#truthy {status = "done"}:

WITH link(name) = """https://www.example.com/images/$name.png""":

multiProgressBarWithImage(
  ARRAY(todo, inProgress, done), COUNT{1},
  ARRAY("Red", "Orange", "Green").MAP(link), link("Gray"),
  200
)
```

Multi-bar with Numbers

In this last example, the status bar displays an issue count for each status, when the bar width permits. This code could be easily customized to display either the actual number of issues or their percentage.

Progress bar with numbers

```
//Parameters: granularity - length of bar-chart in characters; bar - filler of the
bar chart
WITH multiProgressBarWithNumbers(progressArray, maxProgress, colorsArray,
colorForRemaining, granularity, bar) = (
  WITH bars(count, value, color) = (
    IF count <= 0:
      ""
    ELSE:
      WITH bars = (
        WITH charsForValue = LEN(value):
        IF count >= charsForValue + 2:
          WITH charsBeforeValue = FLOOR((count - charsForValue) / 2):
          REPEAT(bar, charsBeforeValue)
          CONCAT value
          CONCAT REPEAT(bar, count - charsBeforeValue - charsForValue)
        ELSE:
          REPEAT(bar, count)
      ):
      """{color:$color}$bars{color}"""
  ):
  WITH barCounts = progressArray.MAP(FLOOR($ / maxProgress * granularity)):
  progressArray.INDEXES()
    .MAP(bars(barCounts.GET($), progressArray.GET($), colorsArray.GET($)))
    .MERGE_ARRAYS(bars(MAX(0, granularity - barCounts.SUM()), MAX(0, maxProgress -
progressArray.SUM()), colorForRemaining))
    .JOIN("", "", "")
):

WITH todo = COUNT#truthy {status = "to do"}:
WITH inProgress = COUNT#truthy {status = "in progress"}:
WITH done = COUNT#truthy {status = "done"}:

multiProgressBarWithNumbers(
  ARRAY(todo, inProgress, done), COUNT{1},
  ARRAY("red", "orange", "green"), "gray",
  20, "▌"
)
```

Simple Burn-down Chart

You can get even more creative and use wiki markup to build mini-charts – including this simple burn-down chart. In this example, our chart displays created issues in red and resolved issues in green, with each pair corresponding to one day in a week.

Due to space limitations, there is a height limit of 20 pixels imposed within the chart, but this is more than enough to create a simple, powerful visualization.

Burn-down chart

```
WITH dataArray = ARRAY(
  COUNT#truthy {DATE_SUBTRACT(NOW(),6,"days") <= created  and DATE_SUBTRACT(NOW(),5,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),6,"days") <= resolved and DATE_SUBTRACT(NOW(),5,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),5,"days") <= created  and DATE_SUBTRACT(NOW(),4,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),5,"days") <= resolved and DATE_SUBTRACT(NOW(),4,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),4,"days") <= created  and DATE_SUBTRACT(NOW(),3,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),4,"days") <= resolved and DATE_SUBTRACT(NOW(),3,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),3,"days") <= created  and DATE_SUBTRACT(NOW(),2,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),3,"days") <= resolved and DATE_SUBTRACT(NOW(),2,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),2,"days") <= created  and DATE_SUBTRACT(NOW(),1,"
days") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),2,"days") <= resolved and DATE_SUBTRACT(NOW(),1,"
days") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),1,"days") <= created  and DATE_SUBTRACT(NOW(),8,"
hours") > created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),1,"days") <= resolved and DATE_SUBTRACT(NOW(),8,"
hours") > resolved},
  COUNT#truthy {DATE_SUBTRACT(NOW(),8,"hours") <= created},
  COUNT#truthy {DATE_SUBTRACT(NOW(),8,"hours") <= resolved}
):


//25 is maximum working height
WITH maxHeight = 25:
WITH maxValue = dataArray.MAX():
WITH getPicture(index) = (IF index.MOD(2) == 0: "https://www.example.com/images/
Red.png" ELSE: "https://www.example.com/images/Green.png"):
WITH getHeight(index) = IF maxValue : FLOOR(dataArray.GET(index) / maxValue *
maxHeight) ELSE : 0 :

IF itemtype != "issue":
dataArray
  .INDEXES()
  .MAP("""!${getPicture($)}|height=${getHeight($)},width=5!""")
  .JOIN("", "", "")
```

The criteria for issue inclusion can be easily customized to your team's needs. As mentioned above, we recommend hosting image files locally.

3.8.3 Formula Reference Documentation

For an introduction to Formulas and the Expr language, see Formulas(see page 228).

3.8.3.1 Expr Advanced Reference

The Expr language is used in the Formula columns and in other places in Structure to produce calculated values based on Jira and Structure data. This reference is a detailed description of the Expr language, its syntax and the calculation rules.

To start learning Expr and what you can do with Structure Formulas, check out Sample Formulas(see page 263) or a more high-level Expr Language description(see page 235). This is an advanced material that may be useful for digging deeper, troubleshooting, or building advanced formulas. It may require some programming knowledge.

This reference contains railroad diagrams to illustrate some of the language syntax. They are intentionally simplified and should not be considered a full definition of the language.

Introduction

In its simplest form, Expr language is built to be similar to Excel or Google Sheets formulas. Similarity to spreadsheet formulas, when possible, was a design goal, particularly when performing arithmetic operations, using functions, and referencing values from the spreadsheet (Jira fields or results of other formulas). The language becomes more complex once you need to work with arrays or items with properties.

Expr is a declarative, dynamically typed language with elements of functional programming.

A few properties of the language are:

- Expr is case-insensitive. Two identifiers different only in the upper or lower case will mean the same thing.
- Whitespace, including new lines, is not meaningful. It is only required to separate word operators and identifiers; in all other cases there can be an arbitrary number of whitespace symbols.
- Except in Text literals, the language supports only English (ASCII) letters, numbers and some punctuation symbols.
- Aggressive type conversion: when a function requires a certain type and the value passed is of a different type, Expr will make the best effort to convert the passed value to the required type.

Expressions and Values

A program written in Expr language is *an expression*. An expression is evaluated by Structure and produces a *value.* In a Formula column, the expression is calculated for each row by applying the expression to the issue or another item in that row, and the result is displayed in the corresponding Formula column cell.

Expr language is *composable*. If you have a valid expression, you can include it as a part of some other expression and it will be valid. Sometimes that will require wrapping it with parentheses.

Lazy Calculation

Expr expressions are calculated in a lazy manner (excluding aggregate functions and embedded queries). This means that, for example, only one branch in an IF expression is going to be calculated.

Displaying Calculation Result in a Formula Column

The Formula Column(see page 463) is the primary means of authoring and using formulas. It currently can display simple values, items, and flat arrays. (See Values and Types below.) If the formula returns a more complex construct – nested arrays or key-value maps, the current version of the column will not be able to display it. However, it doesn't mean the formula is not getting calculated!

Comments

At any place where a formula allows whitespace, you can use comments. Comments can span multiples lines or just one.

- Multi-line comments start with "`/*`" and end with "`*/`" and can span multiple lines. Multi-lined comments cannot be nested.
- Single-line comments start with "`//`" and continue through the end of the line.

Identifiers

All named elements of Expr – variables, local variables, functions and others – must have a valid identifier as their name.

An identifier consists of letters (Latin alphabet only: a-z, A-Z), digits (0-9) or underscore (_) characters. The first character must be a letter or an underscore. An identifier must not be a keyword.

> Versions 1.x of Expr also allowed the period character (`.`) to be a part of the identifier. This possibility has been removed in Expr 2.0.

Keywords

The following keywords must not be used as identifiers.

`AND, CONCAT, ELSE, IF, NOT, OR, UNDEFINED, WITH`

Expr Language Versions

Expr Version	Structure Version	Major Improvements
Expr 1.0	Structure 4.0	Original version introduced
Expr 1.5	Structure 4.2	Local variables, aggregate functions, comments
Expr 2.0	Structure 7.0	Arrays, Items and Properties, User Functions, Chained Function Calls, Embedded Queries, Conditional Expression, Text Snippets, Concatenation Operator

For the compatibility notes in Expr 2.0, see Changes to Expr in Structure 7(see page 419).

Values and Types

All expressions, when evaluated, produce either a value or an error. All values are immutable – if you need to change a value, you derive another value from it.

Each value belongs to a certain type. There are simple and complex types.

Simple types represent just one value and allow you to write literal values (constants). Simple types are:

- Undefined
- Text
- Number

Complex types represent a value that has some internal structure. Complex types are:

- Item
- Array
- User Function
- Key-Value Map

Errors are special values, but they usually cause the whole expression to return an error.

A library function may expects certain types as parameters and will produce a particular type as the result. Expr will also try to convert one type to another as needed.

Undefined

Undefined is a special value, which represents "no value" or a missing value. For example, variable `Assignee` will have undefined value if the issue is unassigned.

Undefined value can be used instead of a value of any other type.

To use the undefined value in a formula explicitly, you can write **`undefined`**.

Functions can return **`undefined`** when the result of the function is not specified. For example, the function **`IF(N = 0; "No apples"; N = 1; "One apple")`** only has a specified value when N is equal to 0 or 1. If N is equal to anything else, it returns **`undefined`**.

Text

A text value consists of 0 or more Unicode symbols. Its literal representation consists of the value enclosed in single quotes (') or double quotes ("). Example: **`"Major"`** represents text value *Major*. Similarly, **`'Major'`** represents the same text value.

If the text value itself contains quotes, you'll need to insert a backslash (\) before them. Example: **`"Charlie \"Bird\" Parker"`** represents the text value *Charlie "Bird" Parker*. Alternatively, you can use another kind of quotes to enclose the literal representation: **`'Charlie "Bird" Parker'`**.

If you need to use the backslash at the end of text value, you will need to insert another backslash before it. Example: **`"C:\Users\John\\"`** represents text value *C:\Users\John*.

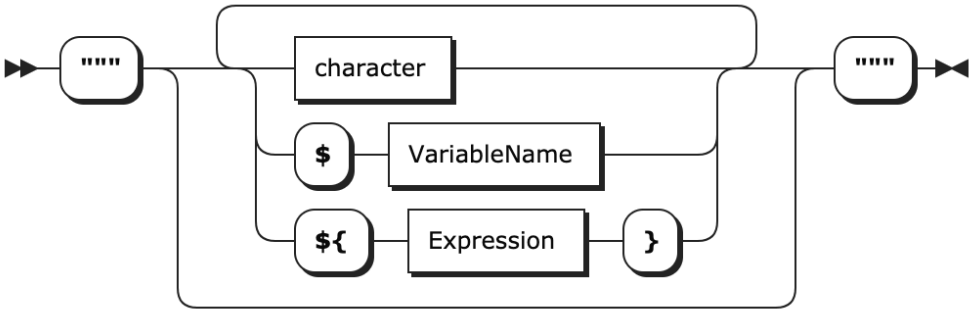
Text Snippets (""")

Sometimes you need a long text value that contains some words and symbols, but also the value of a certain expression or a variable. One way to create it is to use the **`CONCAT`** function and join all the pieces of text together. A more convenient way could be writing a text snippet.

A text snippet starts with triple double quotes (**`"""`**) and continues until the next triple double quotes. It includes ends of line, so you can create multiline text values with a text snippet.

Text snippets may contain special sequences that get replaced with a calculated value:

- A dollar sign followed by a variable name is replaced by that variable's value. For example, **`"""Assigned to: $assignee"""`**.
- A dollar sign followed by an Expr expression in curly braces is replaced by the value of that expression. For example, **`"""Total score: ${ score + subtasks.score.SUM() }"""`**.



Number

A number represents a single numerical value. Aside from representing some quantity, a number value can also represent a point in time or a duration of time. In this case, you can use Format settings in the Formula column(see page 229) to properly display the results as dates or durations.

There are two forms of literal representations of numbers:

- a whole number: **`42`**
- a fractional number: **`0.239`**

Note that only dot (.) can be used as a decimal separator. Comma (,) is used to delimit function arguments. Thus, **`MAX(X, 0,618)`** will be understood as the maximum of three values: *X*, 0, and 618.

Group separators are not supported, so **`100 000`** is not a literal representation of number 100000. However, you can write a string value and explicitly call the conversion function – **`NUMBER("100 000")`**. See number conversion(see page 296).

> **Technical note**: internally, numbers are represented as decimal floating-point numbers with 16 digits of precision and half-even rounding. Most of the operations are carried out in this form; however, some of the more sophisticated functions, such as `SQRT`, might first convert the numbers into binary floating-point, calculate the result and then convert it back into decimal floating-point.

Date and Date/Time

A Date/Time value is represented as an integer number of milliseconds since Unix epoch (1970-01-01 00:00:00 GMT). Date functions will help you convert those numbers to readable Text values.

A Date value (without the time component) is represented in the same way as the date/time value, calculated for the midnight of the specified day – *in the user's current time zone.* ⚠️ This means that expressions involving Date values (for example, using Jira's Due Date field) may produce different results for different users if they are in different time zones.

Duration

A Duration value is represented as an integer number of milliseconds. Duration functions help transform duration values into readable Text values.

Boolean

A boolean value, such as returned from comparison functions, is represented by a number `1` if true, or `0` if false.

Additionally, all values can be converted to a boolean value based on whether they are "truthy". (See Value Conversions.)

Item

An item value represents an object that you can potentially have as a row in a structure – a Jira issue, project, sprint, version, user, status, priority and others. It can also represent purely Structure-owned objects like folders.

Items have properties that you can access by writing **`item.propertyName`**, or **`(item-resulting expression).propertyName`**. The types of items, their properties and their type is listed in Item Property Reference(see page 406).

Special conversion of Item values

Item values always can be used as Text. Each item will then be represented by a text – an issue by issue key, a project by its name, etc.

Also, items may be compared using `UMAX` and `UMIN` functions, which will respect the "natural" order for the items. Most items will be compared alphabetically, but some will have a predefined order, like Priority items.

Special treatment of Item values by generators

Similar to max/min operations, if a formula column produces Item values, then you can sort by that column (or use Attribute Sorter) to reorder the structure according to the natural order of the resulting items.

If an Attribute Grouper is configured with a formula that produces Item values, the groups will become those items.

Array

An array value is a list of other values. Each value in an array is called an element of the array.

Each value in an array may be of any type – you can have an array of texts, array of items, array of arrays, and so on. An array can contain different types of values in each element – the first one could be a number, the second one an item, and the third one an array of some other elements.

The values in an array may be accessed by their index. The first value has index **0**, the second value has index **1**, and so on.

There's a number of array functions that allow you to write formulas that deal with multiple values – see Array Functions(see page 345).

Creating an array

You can create an array by calling the **ARRAY** function: **`ARRAY(element1, element2, element3, ...)`**

When you need to change an array, you apply some transformations to an existing array – most frequent being **FILTER** and **MAP**. This effectively creates a new array.

Accessing array elements

You can access an element of an array by its index, using the **GET** function: **`array.GET(index)`**

However, it's typically not needed, as the calculations with arrays are done mostly with **FILTER**, **MAP** and **REDUCE**.

Special treatment of Array values by the Attribute Grouper generator

When an array value is returned by a formula to the Attribute Grouper generator, it will create multiple groups – one for each non-empty, distinct value returned by the formula.

User Function

A user function value (sometimes called a "lambda value") represents a piece of Expr code that defines a function – see User Functions below.

Typically, you create a user function value to apply it to an array via one of the system functions, such as **FILTER** or **MAP**.

For example, consider the following code:

```
worklogs.FILTER($.author = ME()).MAP(w -> w.timeSpent).REDUCE((a, b) -> a + b)
```

There are three user functions here, **`$.author = ME()`**, **`w -> w.timeSpent`**, and **`(a, b) -> a + b`**. The first defines the condition for filtering work logs, the second defines how to transform each work log into a number, and the third one defines how to aggregate multiple numbers into one.

Each of these three pieces of Expr are evaluated and are represented by a value of the "User Function" type, and passed to FILTER, MAP, and similar functions.

As with any other value, a User Function value can be assigned to a variable. A more traditional function definition form could be used for that. The following two definitions are identical:

```
WITH square = x -> x * x :
...
WITH square(x) = x * x :
...
```

Key-Value Map

A key-value map is produced by some of the Expr functions, and is a collection of pairs, where "key" is the property name and "value" is the property value.

Currently, only **GROUP** function creates a key-value map, with key `group` mapping to the value the array is grouped by, and key `elements` mapping to an array of the matching elements.

To access the value, you need to write **kvm.keyName**, where **kvm** is a local variable or an expression that represents the key-value map.

Keys are supposed to be well-known; there's no way to iterate over all keys.

There's no way for the user to create an arbitrary Key-Value Map; this type is intended only for some system functions.

Errors

Errors are special values, which indicate that the calculation of some part or the whole of the expression encountered a problem.

The list of possible errors is available at Expr Error Codes(see page 414).

Normally if an error occurs somewhere in the formula, the result of the whole calculation is an error as well. But, if receiving an error from a part of the expression is legitimate, you can use the **IFERR** and **ISERR** functions to handle it.

Value Conversions

Each Expr function expects a value of a certain type to be passed as each of its parameters. (See Expr Function Reference(see page 345).) When a value of a different type is encountered, an automatic conversion is attempted. If conversion is not possible, an error will result.

The conversion rules follow some basic principles:

- Best effort is made to convert simple types, such as converting a text to a number
- An item can be converted to text that represents it
- Any value can be converted to an array of one element (array "wrapped around" a value)
- An array with just one value can be converted to that value ("unwrapping" the array)

There are certain more specialized cases related to Text/Joined parameters, parameters marked as /Each and parameters of the User Function type.

The table below summarizes all the conversion rules.

Actual type → **Required type ↓**	**Number**	**Text**	**Item**	**Array**	**Key-Value Map**	**undefined**
Number	pass as is	• empty string → undefined • try to convert to a number • if unsuccessful: error	use the item's text representation, according to the item type, and then use the rules for converting Text to Number	• empty array → undefined • has only one element → try to convert that element • otherwise: error	error	undefined
Integer	• use if an integer value • otherwise: error	*same as above, and try to convert to integer*				
Date	• same as above: timestamps are represented as "unix epoch milliseconds" • dates are represented as a timestamp of the corresponding day's midnight in the user's time zone • using non-integer values may result in error					
Boolean	• false if zero • true otherwise	• false if empty or only whitespaces • true otherwise (including "0"!)	true	• false if empty • true otherwise (including an array with 0 as an element)	true	false

Text	convert to text	pass as is	use the item's text representation, according to the item type	• empty array → undefined / empty text • has only one element → try to convert that element • otherwise: error	error	undefined / empty text
Text/ Joined	*same as above*			• collect a **Text** (not a Text/Joined!) value for each non-undefined element • join elements into one string with ", " as a separator	*same as above*	
Array	array with one element	array with one element	array with one element	pass as is	array with one element	undefined / empty array
Item	error	error	pass as is	• empty array → undefined • has only one element → try to convert that element • otherwise: error	error	undefined
Key-Value Map	error	error	error	• empty array → undefined • has only one element → try to convert that element • otherwise: error	pass as is	undefined
Any	any value works					

Text to Number Conversion

Some functions expect their arguments to be number values. In case an argument is a text value, we try to interpret it as a number. This can be useful if the value comes from a variable that represents a text custom field, which contains numbers — e.g., imported from some external system.

If conversion is successful, that number is used as the value for that argument. If conversion is not successful, functions can either produce an error, ignore that argument, or substitute some default — it depends on the function; see Expr Function Reference(see page 345) for details.

The first step is to accommodate for variations in number formatting. Conversion supports these formatting symbols:

- Decimal fraction separators: comma (`,`), dot (`.`)
- Digit group separators: comma (`,`), dot (`.`), apostrophe (`'`), space (`␣`)

Conversion expects that the text contains 0 or 1 decimal mark, and 0 or more group separators of the same kind. If the text contains any other formatting symbols, conversion fails. Decimal mark must come after all group separators, otherwise conversion fails.

If the text contains only one formatting symbol, and it's a dot (`.`), it is always treated as a decimal mark. If the text contains only one formatting symbol, and it's a comma (`,`), then it is treated as a decimal mark if a comma is used as a decimal separator mark in the Jira default language[52]; otherwise, it is treated as a group separator. For instance, if the default Jira language is English, "**`101,112`**" will become 101112, whereas if it is German locale, it will be 101.112. And regardless of language, "**`1 100,23`**" will become 1100.23: space is interpreted as a group separator, and comma can only be the decimal fraction separator here.

If the group separator is a dot (`.`), then all groups except the first one must have 3 digits; otherwise, conversion fails.

After determining decimal mark and group separator symbols, conversion removes all group separator symbols and replaces the decimal mark with a dot. Note that if text contains several whole numbers separated by spaces, conversion will think it is one number, for example, "**`10 11 12`**" will become 101112. Similarly, "**`10,11,12`**" will become 101112.

The final step of conversion is to recognize the resulting text as either Expr's literal number representation or scientific or engineering notation. Examples:

```
0.239
-1.32e5
12e-3
```

Conversion to Boolean: Falsy and Truthy Values

A value is *falsy* if it is:

- `undefined`,
- number 0,
- an empty text value (`""` or `''`), or a text value that contains only space characters,
- an empty array.

All other values are *truthy.*

When converting to a Boolean, truthy values become true and falsy values become false.

By convention, when functions or logical operators need to construct a truthy value, they use the number `1`.

Text vs. Text/Joined

When a function declares that it requires "Text/Joined" value, it means that the value will be converted to a text, with an additional special handling of the array type:

- If it's an empty array, or an array with only one element, the conversion will be the same as for "Text" type.
- If an array with multiple values is passed, then a) each element of the array will be converted to a Text value, b) all these texts will be joined together with a comma as a separator.

Here's an example illustrating the difference when fixVersion is passed as a parameter - notice that in the third row, there are multiple values in fixVersion (because it's an array), so Text and Text/Joined are treated differently:

52 https://confluence.atlassian.com/adminjiraserver071/choosing-a-default-language-802592304.html

fixVersion	Function accepting Text will receive	Function accepting Text/Joined will receive
(no value)	undefined	undefined
v1	"v1"	"v1"
v1, v2	error	"v1, v2"

Passing Implicit User Function as a Parameter

You can always pass an implicit User Function (the one containing the "$" symbol) as an argument to a function. This will result in the call to this function to become an implicit User Function value itself.

For example, consider the following expression:

```
fixVersions.FILTER(YEAR($.releaseDate) = 2021)
```

Function **`YEAR()`** expects a date, but it receives a User Function instead (**`$.releaseDate`**), which produces the release date for each passed version. As a result of applying **YEAR()** to that User Function, we will get another user function, which produces the *year of the release date* for each passed version.

This logic applies only to the implicit User Functions, defined with the $ sign.

The functions that expect a User Function parameter, like FILTER, are exclusions from this rule.

Variables

Variables (also known as free, or externally set variables) represent some values that will be fed into the formula for each Structure row that the formula will be applied to.

For example:

```
IF priority = "Blocker" : parent.estimate + x
```

In this formula, "priority", "parent" and "x" are all variables – they vary from one row to another. (They will not change the value while calculating the expression for a single row.)

There's no need to declare a variable, you can immediately start using it. All valid identifiers that are used like variables will be treated as such.

Each formula is expected to contain at least one variable – otherwise, the result will be the same for each row.

Variable to Attribute Mapping

Each variable should be mapped to a valid attribute – such as a Jira field or a Structure attribute, so when an expression is calculated for a particular item, the value of that attribute becomes the variable's value.

If you use one of the well-defined variable names, it will be automatically mapped to the corresponding attribute. See Standard Variable Reference(see page 402).

If you use an arbitrary variable name, such as "x", you will need to map it as described on the Mapping Variables(see page 232) page.

If a variable is not mapped, or if the item does not support the mapped attribute, the value of the variable will be the `undefined` value.

"this" Variable

One of the well-defined variable names is "this". It is mapped to an attribute that provides a value of Item type, representing the item for which the formula is being calculated.

This may come in handy in certain cases. For example, to analyze issue links and pick the "other side" of a link, regardless of whether it's an incoming or an outgoing link:

- **`issueLinks.MAP(IF $.source = this : $.destination ELSE $.source)`**

Alternatively, you can use "item" with the same meaning.

Local Variables

Local variables are similar to Variables, but they are not mapped to an item's attribute or Jira field, but rather defined and calculated right in the expression.

The declaration syntax is the following:

Note the colon ("**`:`**") that separates the expression assigned to the variable and the expression where the variable is used.

A few facts about local variables:

- *`ExpressionWithLocalVariable`* may start with another local variable definition, so you can introduce many local variables in sequence. When defining a second variable, you can use the first variable already defined, and so on.
- Local variables can "shadow" previously defined local and free (mapped) variables with the same name. If you write **`WITH priority = 10: <expression>`**, then when calculating `<expression>`, the value of **`priority`** will be 10, even if there was a variable attached to the issue's priority in the enclosing scope.
- The **`WITH...`** construct is itself an expression, so you can use it, enclosed in parentheses, anywhere an expression can be used. The name defined in this expression is not visible outside the **`WITH...`** expression.

Immutability

Expr language constructs are immutable. Once a local variable is defined, it cannot change its value. (So, in fact, calling it a "variable" is not exactly correct. Although, if a local variable depends on external variables, which vary from item to item, the local variable itself will also vary from item to item.)

So if you're building your formula and you need to take a number of values through a series of calculations, you may need to use multiple local variables, going through each step and assigning each intermediate result to another local variable.

Operators

Operators allow writing formulas in a convenient way using traditional logical and arithmetic operations. Each operator has one (unary operators) or two operands. Each operand could be a variable, a number, or just any expression, which sometimes will need to be in parentheses.

The operators translate into calling the corresponding functions on operands as arguments. If an operand expression evaluates to an error, then the operator's result will also be an error.

Available operators are:

Operator(s)	Symbol(s)	Priority	Type of Operands	Result Type	Corresponding Function(s)
Logical negation	`NOT`	7	Any	Number (Boolean)	`NOT`
Unary sign	`+ -`	7	Convert to Number	Number	`SUM, MINUS`
Multiplication and division	`* /`	6	Convert to Number	Number	`MUL, DIV`
Addition and subtraction	`+ -`	5	Convert to Number	Number	`SUM, MINUS`
Concatenation	`CONCAT`	4	Convert to Text/ Joined	Text	`CONCAT`
Equality/Inequality check	`= !=`	3	Any	Number (Boolean)	`EQ, NE`
Numeric Comparison	`< > <= >=`	3	Number	Number (Boolean)	`LT, GT, LE, GE`
Logical AND	`AND`	2	Any	Any(*)	`AND`
Logical OR	`OR`	1 (lowest)	Any	Any(*)	`OR`

The operators are listed in their priority order. The priority is important when an expression could allow different interpretations about which operators are applied first.

For example, in an expression **`progress + parent.progress * weight < threshold AND priority != "Blocker"`**, the multiplication is executed first, then the addition, then the comparisons, and then the logical AND. If you'd like to alter the order of operator application, use the parentheses.

Logical Negation (NOT)

To negate a logical value or expression, use the **`NOT`** operator. Instead of **`NOT`**, an exclamation mark (**`!`**) can also be used.

The operator produces **`0`** if the operand is a truthy value, and **`1`** otherwise. Therefore, it may be applied to any value.

If the value negated is an expression with other operators, it should be contained in parentheses.

Examples:

- **`NOT resolved`**
- **`NOT (storyPoints > 0 AND storyPoints < parent.maxStoryPoints)`**

Unary Sign (+ -)

The operator first attempts to convert the value of an expression to a number. If conversion succeeds, + produces this number, and − produces the negated number.

If the conversion to a number fails, and the value of the expression is falsy, the negation produces `undefined`. Otherwise, it produces an error.

Arithmetic operators (* / + -)

Arithmetic operators are: addition (+), subtraction (-), multiplication (*) and division (/).

Multiplication and division have precedence over addition and subtraction.

These operators convert their arguments to numbers. A non-empty, non-number argument would produce an error. Falsy non-number values are treated as zero.

Examples:

- **"" + 1 → 1**
- **"foo" + 1 → error**
- **"" * 1 → 0**
- **"foo" * 1 → error**
- **"" - 1 → -1**
- **1/0 → error**

If any subexpression produces an error, the operator produces the same error.

Concatenation (CONCAT)

The concatenation operator converts each operand to a text value (Text/Joined, to be precise) and creates a new text by joining them together. It is identical to calling the CONCAT function on its operands:

```
value1 CONCAT value2 CONCAT value3 = CONCAT(value1, value2, value3)
```

The operator can be used to increase the readability of a formula.

Note that CONCAT is applied after all the arithmetic operators but before comparisons and logical operators.

Equality check (= !=)

The "equals" operator (=) checks that both arguments are "essentially the same value". The "not equals" operator (**!=** or **<>**) produces the inverse value.

The comparison rules are based on the type of the values compared.

Equality operator will return true (1) if any of the following conditions hold:

- Both values are **Undefined.**
- One value is a **Number**, and the second value is the same number, or can be converted to the same number.
- One value is a **Text**, and the second value can be converted to Text (using Text/Joined), and both values are "essentially the same".
    - The differences in letter forms and leading and trailing whitespace are ignored (thus **" cote " = "côte"**).
- One value is an **Item**, and the other value is the same item. (Note that if the other value is a Text, both values can be compared as text values.)
- One value is a **Key-Value Map**, and the other value is also a key-value map with the same contents (these equality rules applied to all elements).

- One value is a **User Function**, and the other value is exactly the same user function (not just having the same logic).
- One value is an **Array**, and either of the following is true:
  - The other value is also an **array**, has the same number of elements and the elements are respectively equal.
  - The other value is **undefined** and the array does not contain any non-undefined elements.
  - This array contains only **one element** and this element is equal to the other non-array value we are comparing the array with.

In all other cases, the equality returns false (`0`).

Note that you can compare an item to a text, for example,

```
IF project = "My Project" : ...
```

The item will get converted to a text using value conversion rules described earlier.

If one value is a number and the other value can be converted to a number, both values are treated as numbers. However, if both values are text, they will be treated as text, even if both can be converted to a number. You can use the NUMBER(see page 0) function to force a value to be numeric.

- `3.4 = 3.40 → 1`
- `3.4 = "3.40" → 1`
- `"3.4" = "3.40" → 0`
- `NUMBER("3.4") = "3.40" → 1`

Numeric Comparison (< > <= >=)

The ordering / comparison operators work on numbers only:

- **<** (less than)
- **>** (greater than)
- **<=** (less than or equal)
- **>=** (greater than or equal)

If either of the values is text, the operator attempts to convert it to a number. If the conversion fails, the result is an error.

If any value is undefined, strict operators (**<**, **>**) produce `0`. Non-strict (**<=**, **>=**) produce `0`, unless *both* values are undefined (because they are equal).

Logical operators (AND OR)

The logical operators are used to combine other logical conditions, or to pick an alternative or a conditional value.

- **OR** also can be written as "**||**" or "**|**"
- **AND** also can be written as "**&&**" or "**&**"

When both operands are Number(Boolean), then the operators perform the corresponding boolean operation.

However, you can use these operators with non-boolean operands in a "short circuit" way, based on whether the operands are "truthy" (see above).

- **`a OR b`** – if "a" is truthy, "b" is not evaluated and the result of the operation is "a"; otherwise the result of the operation is "b".

- **`a AND b`** – if "a" is falsy, "b" is not evaluated and the result of the operation is "a"; otherwise the result of the operation is "b".

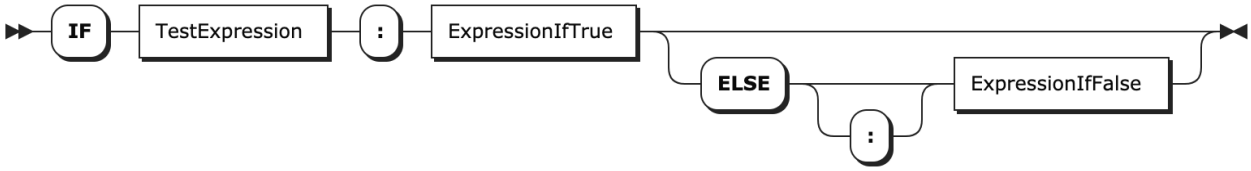
Note that "b" is not evaluated in case the result of the operation is equal to "a". This might be important in case calculating "b" could result in an error.

Examples:

- **`assignee OR "UNASSIGNED"`** – This will produce either the issue's assignee user key or (if the issue is unassigned) the text value "**`UNDEFINED`**".
- **`!assignee AND status = "OPEN"`** – This will produce **`1`** if the issue is unassigned and in status **`OPEN`**, and **`0`** otherwise.
- **`count AND total / count`** – This will produce some average number (total / count), unless count is 0 – in this case the result will be 0. Note that there will be no division by zero error.

Conditional Expressions

A conditional "IF" expression allows you to switch between two expressions, based on whether a condition is true (truthy) or false. It is identical to calling the IF function with two or three arguments.



The "ELSE" part, as well as the colon ("**`:`**") after ELSE are optional. If the ELSE part is omitted, and the test expression evaluates to false, the result is `undefined`.

If you use nested IF expressions with only one ELSE, the ELSE part applies to the innermost IF. We recommend using parentheses to make it clear which IF it applies to.

You can use indentation to make the formula with nested IFs more readable – but the indentation has no effect on how the formula is parsed. Use parentheses!

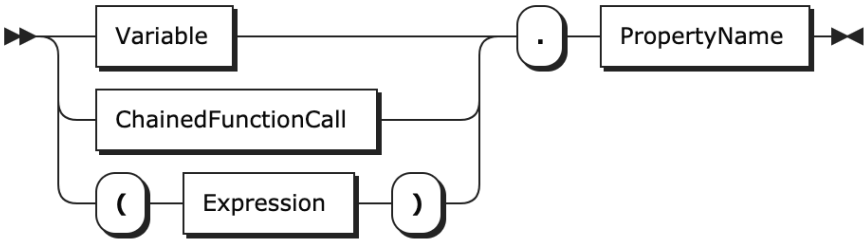
Examples:

```
IF assignee = ME() : "mine!"
---
IF dueDate < NOW() : "overdue!" ELSE: """${DAYS_BETWEEN(NOW(), dueDate) - 1} days
left!"""
---
IF priority = "Critical":
IF dueDate < DATE_ADD(TODAY(), 7, "days"):
   "critical and urgent"
ELSE:
   "critical and not urgent"
```

Property Access

Some Expr values – Items and Key-Value Maps – contain "properties", named values that are parts of the bigger value. For example, a Version item contains properties such as "name", "description", "releaseDate". And function GROUP returns a value that is an array of Key-Value Maps, each map having "group" and "elements" properties.

To access a property, use the dot ("**.**"), followed by the property name. The names are case-insensitive.



Examples:

- **`version.releaseDate`**
- **`sprint.startDate`**
- **`issueType.isSubtask`**

An alternative way to access a property is by using the **`ACCESS()`** function – in this way, the property name itself may be calculated:

- **`parentTask.ACCESS("Story Points")`**

⚠ If the value does not have the requested property, or if the value is neither an Item nor a Key-Value Map, the resulting value is `undefined` – not an error! (See also a note about arrays below.)

ⓘ There's no way to check if a value contains a certain property (other than try and access it), or list all available properties for an item.

Accessing Custom Fields via Properties

In most cases a formula will refer to the issue's custom fields directly by name – for example, the formula **`impact / storyPoints`** uses the values of two custom fields, Impact and Story Points, of the currently calculated issue to calculate the benefit-to-cost ratio.

You can, however, use a formula to access other, related issues, and their corresponding custom fields – for example, the formula **`impact / (storyPoints + subtasks.storyPoints.SUM())`** calculates a similar value but takes into account the cost (in story points) of all the subtasks.

Note that the actual custom field is named "Story Points", with a whitespace. When you access a property of an issue item, Structure tries to match the property name with available custom field names using loose rules, dropping whitespace and any non-identifier-friendly characters. The property name is also case insensitive.

You can also use the ACCESS() function to specify the name of the field precisely, or use "customfield_NNNNN" property name to identify a field by its ID.

- **`parent.storyPoints`**
- **`parent.ACCESS("Story Points")`**
- **`parent.customfield_10000`**

✓ You can use `this` variable to access properties of the currently calculated item. Normally it's not needed, since the same values are available as corresponding variables.

Accessing Property of Each Element in an Array

You can apply property access to an array of values. Expr will then apply property access to each element and return the result as an array.

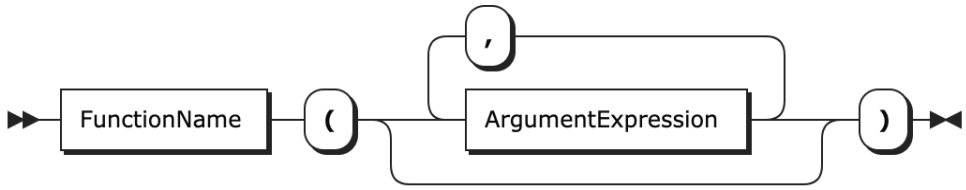
For example:

- **fixVersion.releaseDate** – will return an array of release dates
- **worklogs.author.UNIQUE()** – will return a list of people who logged work
- **subtasks.remainingEstimate.SUM()** – will return a total remaining estimate from the subtasks

In the resulting array, all **`undefined`** results will be removed and the array will be flattened. This is the same behavior as shown by functions with /Each parameter type.

Functions

A function takes zero or more values, and produce another value. A function call consists of a function name (an identifier), followed by its arguments enclosed in parentheses. An argument can be any expression. Different arguments are separated by commas (**`,`**) or semicolons (**`;`**) — for one function call, all separators must be the same.



Examples:

- **`NOW()`**
- **`ROUND(storyPoints / 10)`**
- **`FILTER(sprint, $.state = "active")`**
- **`MAKE_DATETIME(2017; 12; 31; 23; 59; 59)`**
- **`myUserFunction("argument1", "argument2")`**

A function call can evaluate only some or even none of the arguments, depending on the function. This is useful for functions that perform choices. For example, in an IF function, the argument that wasn't chosen is not evaluated, so the whole expression doesn't produce an error when that argument produces an error.

System and User Functions

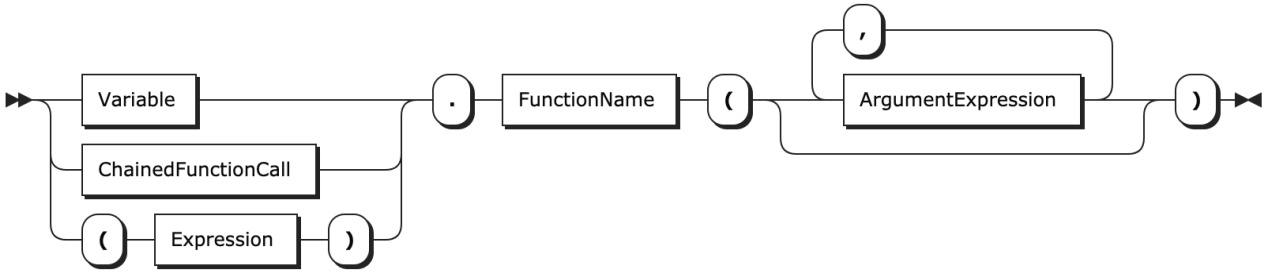
System functions are provided by Structure. The functions are listed in the Expr Function Reference(see page 345). Each function expects a certain number and type of parameters.

User functions are explained in the next section. A user function is called using the same syntax as a system function.

> User functions can take any number of arguments, regardless of how many arguments are declared. If a parameter was declared, but a value was not provided when calling the function, the parameter's value will be `undefined`.

Chained Function Calls

A different way to call a function is by "chaining" it to its first argument by adding a period ("**.**"), a function name, parentheses, and any additional parameters, if any.



For example, **`sprint.FILTER($.state = "active")`** is the same as **`FILTER(sprint, $.state = "active")`**.

This allows nice, readable expressions, where a value is sequentially transformed by applying functions to the result of a previous function call:

- affectsVersion.MAP($.releaseDate - $.startDate).MAX()
- linkedIssues.FILTER(x -> NOT x.resolution).MAP(x -> x.remainingEstimate).SUM()

Unlike some other languages, in Expr any function may be written in the chained syntax, regardless of what the value is.

Applying Functions to Arrays

When a function is applied to an array value (meaning that an array is passed as the first argument of the function), the result may be calculated in a number of ways, depending on which function is called and what type of argument it expects.

You can always apply a function to each element of an array using the MAP function. For example:

- **`array.MAP(SOME_FUNCTION($))`**

The cases below relate to cases where an array is passed directly as an argument: **`SOME_FUNCTION(array)`**.

Using an Array Function

There are special functions that expect an array as their first argument – **`FILTER`**, **`MAP`**, and others(see page 296).

There's no special behavior in this case – an array is expected. In fact, if the value passed is not an array, it will be converted to either an array of one element (containing that value), or an empty array if the value is undefined.

Passing Array as an Argument of Text/Joined Type

If a function declares that it expects "Text/Joined" as an argument, then the system will try to convert an array into a text value. See the "Text vs Text/Joined" section above.

For example:

- **`CONCAT("Versions: ", fixVersion)`** → **`"Versions: v1, v2, v3"`**

Passing Array as an Argument of /Each Type

If a function declares that it expects "Number/Each" or "Text/Each" or any other "/Each" type as an argument, then it would work on that simple type, but if an array is passed, it will apply its logic to each element in that array. The result of calling this function will be an array, where each element is a result of applying the function to the original element.

For example:

- **`UPPER(fixVersion) → ARRAY("V1", "V2", "V3")`**

In addition, when applying a function to an array in this way, the resulting array is "flattened" (elements from any sub-arrays moved to be the elements of the top array) and "compacted" (all undefined elements are removed).

Passing Array to a User Function
You can call a user function and pass an array as an argument. No special handling takes place – just as with a system function expecting an array.

All Other Cases
If a system function does not expect an array, but it is passed as an argument, it will try to convert it to the value type it expects. A one-element array will be converted to its single element and an empty array will be converted to `undefined`. See "Value Conversions" above.

If the conversion is not possible, the result will be an error.

User Functions

User Functions are functional *expressions*, defined by the user. (They could also be called "lambdas".) User functions are helpful when the user needs to apply some repetitive action, or to pass an action to be applied to each item in an array.

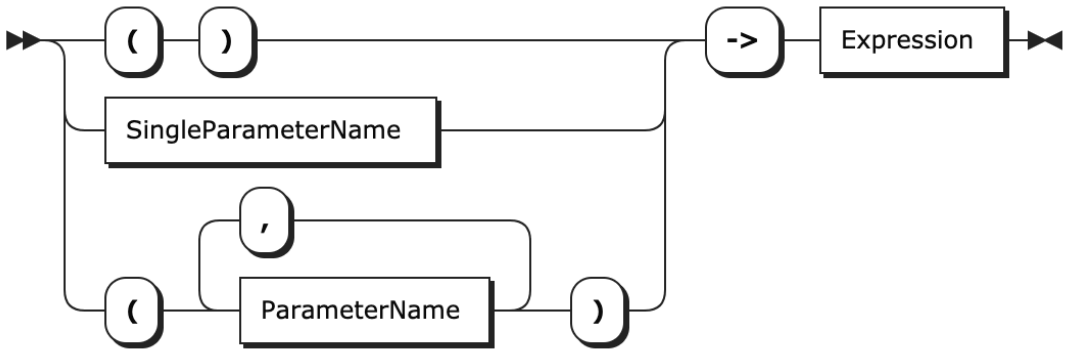
A User Function contains a list of parameters and then an expression that is calculated for these parameters.

A User Function is a *type of value*, so you can assign a user function to a local variable, or pass it to some higher-order function as a parameter.

There are three ways to define a user function.

Functional Expression

A functional expression is the canonical form for user functions. It contains a list of parameters in parentheses, followed by the "maps to" symbol (**->**), followed by the expression calculated by the function. When there's only one parameter, the parentheses can be omitted.



Examples:

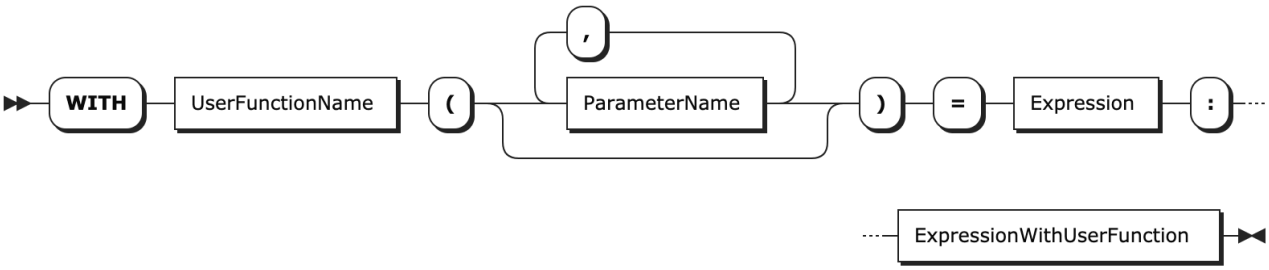
- `() -> START_OF_MONTH(NOW())`
- `(x) -> x * x`
- `version -> version.releaseDate - version.startDate`
- `(s1, s2) -> s1 CONCAT " " CONCAT s2`

All these examples evaluate to a "User Function" value type, which can be assigned to a variable:

- `WITH square = x -> x * x : ...`

Traditional Function Definition

A more traditional function definition looks similar to a variable definition, only the variable is followed by a list of parameters in parentheses, and the expression is based on those parameters.



To rewrite the examples above:

- `WITH currentMonth() = START_OF_MONTH(NOW()) : ...`
- `WITH square(x) = x * x : ...`
- `WITH versionDuration(version) = version.releaseDate - version.startDate : ...`
- `WITH joined(s1, s2) = s1 CONCAT " " CONCAT s2 : ...`

These declarations are identical to the corresponding examples in the previous section, with local variables assigned to the corresponding User Function values.

Implicit Functional Expression ($)

Most of the time when we're creating a formula with an array, we need to apply some kind of operation to each element of the array. Implicit functional expressions help define the corresponding user function easily by having "$" denote "each element".

For example:

- `versions.FILTER($.startDate < NOW())`
- issueLinks.FILTER($.type = "Relates").MAP($.destination)
- `worklogs.UMAX_BY(IF $.author = ME() : $.timeSpent)`

In each case, the expression with "$" is transformed into a User Function with a single parameter, which is then substituted for $. So, the last example from the list above is identical to:

- `worklogs.UMAX_BY(w -> IF w.author = ME() : w.timeSpent)`

When reading these expressions, you can say "each" when the dollar sign is encountered.

⚠ An implicit user function must always be used in an argument to a system function, which expects a user function. Otherwise, it won't be accepted.

For example – here's how we can filter an array to contain only even numbers:

Correct	Incorrect – Parse Error

`ARRAY(1, 2, 3).FILTER(MOD($, 2) = 0)`
... or, alternatively ...
`WITH even(e) = MOD(e, 2) = 0 :`
`ARRAY(1, 2, 3).FILTER(even)`
	`WITH even = MOD($, 2) = 0 :`
	`ARRAY(1, 2, 3).FILTER(even)`

Calling User Functions

If a User Function is assigned to a variable, you can call it in your expression in the same way you call a system function.

- `WITH square(x) = x * x :`
  `square(impact) / square(cost)`

You can also use the chained function call notation:

- `WITH square(x) = x * x :`
  `WITH fquare(x) = x.square().square() :`
  `storyPoints.fquare()`

ⓘ Note that you cannot invoke a functional expression unless it is assigned to a local variable. The following will produce an error: `(x -> x * x)(3)`

Function Name Collisions

Both system functions and user functions are invoked in the same way – **`FUNCTION_NAME(arg1, arg2, arg3, ...)`**, or with a chained call syntax – **`arg1.FUNCTION_NAME(arg2, arg3, ...)`**. This leaves a potential for the user to define a function that has the same name as a system function.

⚠ When Expr encounters a function call, first it looks up if there is a *system* function of that name. ⚠

The system function will be called even if there's a local variable of the same name. To protect the user from name collisions, Expr will show an error if you try to define a function with a name that matches a system function name. (However, it won't be able to detect the collision if a local variable is defined through a series of assignments of a functional expression.)

You *can* define a local variable of any other type with a name identical to a system function's name.

Works as expected	Error

`// Using "SUM" as a local variable,`
`// but "SUM()" is also a system function`
`// and "SUM{}" is an aggregate function.`
`WITH SUM = cost + parent.cost :`
`SUM(SUM, SUM { cost })`
	`// Cannot define a user function with a name collision.`
	`// Note that the language is case-insensitive: "SUM" and "sum" are the same.`
	`WITH sum(issue) =`
	`  issue.timeSpent + issue.parent.timeSpent :`
	`...`

⚠ Note that the function name collision resolution provides potential challenges when upgrading to a newer version of Structure, if that version introduces new system functions.

Let's say you have defined a user function **`LAST_COMMENT()`** in your formula and used it successfully in an older Structure version. If the newer version of Structure adds a system function **`LAST_COMMENT()`**, that formula will likely stop working after the upgrade, and you will need to rename the user function.

To minimize the probability of this happening, we suggest naming your user function in a way that makes potential collision unlikely. It could be a name that is very specific to your configuration, or you can always prepend the name with an underscore – in our example, call it **`_LAST_COMMENT()`**.

Aggregate Functions

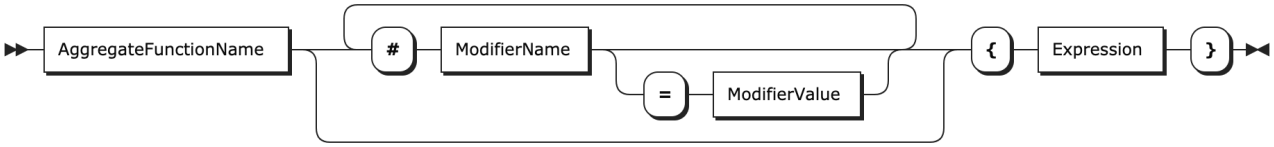
An aggregate function calculates values for some other rows in the structure (for example, for all sub-items), aggregates these values (for example, adds them together), and produces the resulting value to be used in the formula.

For example:

- **`SUM#children { storyPoints }`** – calculates total story points for all the child issues.
- **`PARENT { fixVersion }`** – provides the value of the fixVersion field from the parent's issue.
- **`VALUES { components }`** – collects all distinct components that are set for any of the sub-issues of the current row.

Aggregate functions allow you to calculate complex values that depend not only on the "current" item, but also on other items in relation to it.

An aggregate function starts with a name, optionally followed by modifiers, then curly braces ("**`{}`**"), and inside them – an "inner expression", which will be calculated for some other rows. You can use whitespace between any elements of the aggregate function calls.



The inner expression may return any type of value – number, text, array, and others – *except a user function*. You cannot pass a user function from the inner formula into the outer formula.

Available Aggregate Functions are listed in Aggregate Function Reference(see page 387).

Aggregate Function Modifiers

An aggregate function may have one or more modifiers that govern the aspects of the function's execution. Each modifier starts with hash sign ("`#`"); then comes the modifier's name; optionally followed by the equals sign ("`=`") and a value, which can be a string or a numeric constant. If a value is omitted, it is assumed to be `1` (a representation of *true* in Expr).

Examples:

- `SUM#all { cost }`
- `SUM#all#leaves { IF type = "story": storyPoints }`
- `JOIN#separator=", " { key }`
- `JOIN #separator=", " #fromDepth=0 #toDepth=-1 { key }`

Each aggregate function supports a specific set of modifiers, not all of them. Using an incompatible modifier will result in an error. To learn more about available modifiers and their restrictions, see Aggregation Modifiers(see page 393).

Sharing Values Between Outer and Inner Formulas

It's important to understand that the formula inside an aggregate function – the inner formula – is calculated fully separately from the "outer" formula. Both formulas will share variable mappings (to attributes), but any local variables and user functions defined on one side will not be accessible from the other side.

Will not work!

```
1  // Cannot use "total" on line 4!
2
3  WITH total = SUM#children { storyPoints } :
4  WITH median = MEDIAN#children { storyPoints / total } :
5  ...
```

Correct version

```
1  WITH median = MEDIAN#children {
2    WITH total = PARENT { SUM#children { storyPoints } } :
3    storyPoints / total
4  } :
5  ...
```

As you can see from the example above, you may need to use nested aggregate functions instead.

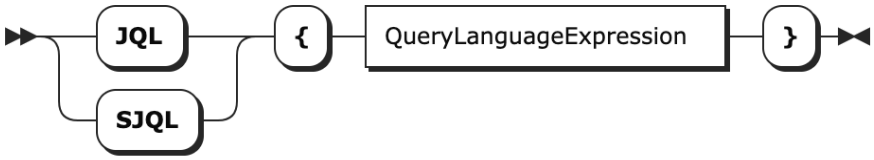
Using Formulas with Aggregate Functions in Generators and Transformations

Note that when you use an aggregate function, it relies on the existing structure to figure out the parent item, child items, and other related items addressed by the formula. When the formula is used to build or transform a structure, the hierarchy and items that the aggregate functions "see" will correspond to the structure that exists immediately before the generator or the transformation is applied.

Therefore, when using a formula with Attribute Filter, Attribute Grouper, Attribute Sorter and other generators, you should apply aggregate functions carefully, understanding what is the preceding structure before the generator is applied. For example, if you use a grouper on a folder with a flat list of issues, the formula in the grouper will see issues and the folder will be their parent item – there are no groups yet!

Embedded Queries

You can embed JQL and S-JQL(see page 428) in a formula to check if the item (the one the formula is being calculated for) satisfies the condition of the query – that is, it will be a part of the query result.



The syntax is similar to calling an aggregate function:

- **`IF JQL { assignee in membersOf("Team-Alpha") } : ...`**
- **`IF NOT SJQL { descendant of folder("Excluded") } : ...`**

The result of evaluating **`JQL {}`** or **`SJQL {}`** is always **`0`** (false) or **`1`** (true).

When SJQL is used, it is always applied to the current structure. (Or the one being generated – see the note below.)

Note that, unlike aggregate functions, these constructs do not use Expr, but rather another other languages, as the inner expression. Use the corresponding documentation as a reference for JQL and S-JQL.

> The embedded queries are calculated separately from the Expr formula they are used in. Therefore, you cannot use any values from the Expr formula inside a JQL or SJQL query or vice versa. Also, you cannot check the query match for any other item except the one the formula is being calculated for.
>
> In other words, the only data that is passed between an embedded query and the outer Expr formula is 1 or 0 depending on whether the current item matches the query.

Using S-JQL in Generators and Transformations

An S-JQL query usually depends on the structure it's being calculated for. So, similar to Aggregate Functions, when you use a formula with an embedded S-JQL in a generator or a transformation, the query will be applied to the underlying "preceding" structure, that exists before the generator is applied.

S-JQL Query Performance

Structure optimizes the calls to embedded queries. A query will be run only once for multiple items that the formula is being calculated for, and the result will be checked separately in the calculations for each row.

That said, the JQL itself may potentially be an intensive calculation, if it uses JQL functions or historical conditions like WAS. Please be careful when trying this JQL in a formula, and watch for how long the value is being calculated before publishing the formula for other users. Normally, calculating a "heavy" formula should not prevent users from doing other things in Structure (including using other columns with formulas), but it can place some stress on the Jira server.

> Please avoid using the **`structure()`** JQL function in JQL or S-JQL that is being embedded in a formula.

Expr Advanced Reference - Introduction

In its simplest form, Expr language is built to be similar to Excel or Google Sheets formulas. Similarity to spreadsheet formulas, when possible, was a design goal, particularly when performing arithmetic operations, using functions, and referencing values from the spreadsheet (Jira fields or results of other formulas). The language becomes more complex once you need to work with arrays or items with properties.

Expr is a declarative, dynamically typed language with elements of functional programming.

A few properties of the language are:

- Expr is case-insensitive. Two identifiers different only in the upper or lower case will mean the same thing.
- Whitespace, including new lines, is not meaningful. It is only required to separate word operators and identifiers; in all other cases there can be an arbitrary number of whitespace symbols.
- Except in Text literals, the language supports only English (ASCII) letters, numbers and some punctuation symbols.
- Aggressive type conversion: when a function requires a certain type and the value passed is of a different type, Expr will make the best effort to convert the passed value to the required type.

Expressions and Values

A program written in Expr language is *an expression*. An expression is evaluated by Structure and produces a *value.* In a Formula column, the expression is calculated for each row by applying the expression to the issue or another item in that row, and the result is displayed in the corresponding Formula column cell.

Expr language is *composable*. If you have a valid expression, you can include it as a part of some other expression and it will be valid. Sometimes that will require wrapping it with parentheses.

Lazy Calculation

Expr expressions are calculated in a lazy manner (excluding aggregate functions and embedded queries). This means that, for example, only one branch in an IF expression is going to be calculated.

Displaying Calculation Result in a Formula Column

The Formula Column(see page 463) is the primary means of authoring and using formulas. It currently can display simple values, items, and flat arrays. (See Values and Types below.) If the formula returns a more complex construct – nested arrays or key-value maps, the current version of the column will not be able to display it. However, it doesn't mean the formula is not getting calculated!

Comments

At any place where a formula allows whitespace, you can use comments. Comments can span multiples lines or just one.

- Multi-line comments start with "`/*`" and end with "`*/`" and can span multiple lines. Multi-lined comments cannot be nested.
- Single-line comments start with "`//`" and continue through the end of the line.

Identifiers

All named elements of Expr – variables, local variables, functions and others – must have a valid identifier as their name.

An identifier consists of letters (Latin alphabet only: a-z, A-Z), digits (0-9) or underscore (_) characters. The first character must be a letter or an underscore. An identifier must not be a keyword.

> Versions 1.x of Expr also allowed the period character (`.`) to be a part of the identifier. This possibility has been removed in Expr 2.0.

Keywords

The following keywords must not be used as identifiers.

`AND, CONCAT, ELSE, IF, NOT, OR, UNDEFINED, WITH`

Expr Language Versions

Expr Version	Structure Version	Major Improvements
Expr 1.0	Structure 4.0	Original version introduced
Expr 1.5	Structure 4.2	Local variables, aggregate functions, comments
Expr 2.0	Structure 7.0	Arrays, Items and Properties, User Functions, Chained Function Calls, Embedded Queries, Conditional Expression, Text Snippets, Concatenation Operator

For the compatibility notes in Expr 2.0, see Changes to Expr in Structure 7(see page 419).

Expr Advanced Reference - Values and Types

All expressions, when evaluated, produce either a value or an error. All values are immutable – if you need to change a value, you derive another value from it.

Each value belongs to a certain type. There are simple and complex types.

Simple types represent just one value and allow you to write literal values (constants). Simple types are:

- Undefined
- Text
- Number

Complex types represent a value that has some internal structure. Complex types are:

- Item
- Array
- User Function
- Key-Value Map

Errors are special values, but they usually cause the whole expression to return an error.

A library function may expects certain types as parameters and will produce a particular type as the result. Expr will also try to convert one type to another as needed.

Undefined

Undefined is a special value, which represents "no value" or a missing value. For example, variable `Assignee` will have undefined value if the issue is unassigned.

Undefined value can be used instead of a value of any other type.

To use the undefined value in a formula explicitly, you can write **`undefined`**.

Functions can return **`undefined`** when the result of the function is not specified. For example, the function **`IF(N = 0; "No apples"; N = 1; "One apple")`** only has a specified value when N is equal to 0 or 1. If N is equal to anything else, it returns **`undefined`**.

Text

A text value consists of 0 or more Unicode symbols. Its literal representation consists of the value enclosed in single quotes (') or double quotes ("). Example: **`"Major"`** represents text value *Major*. Similarly, **`'Major'`** represents the same text value.

If the text value itself contains quotes, you'll need to insert a backslash (\) before them. Example: **`"Charlie \"Bird\" Parker"`** represents the text value *Charlie "Bird" Parker*. Alternatively, you can use another kind of quotes to enclose the literal representation: **`'Charlie "Bird" Parker'`**.

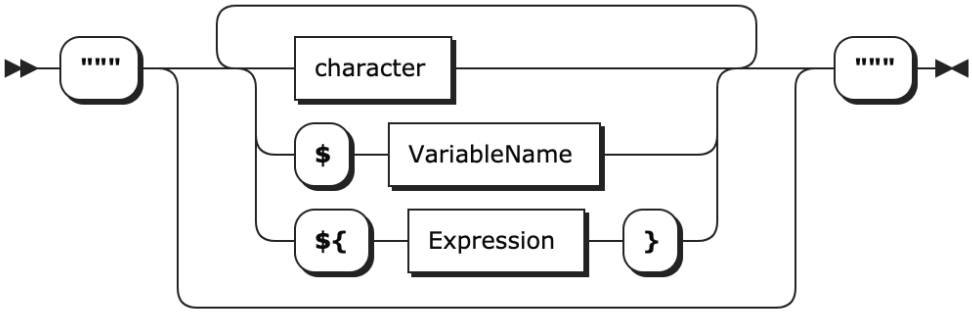
If you need to use the backslash at the end of text value, you will need to insert another backslash before it. Example: **`"C:\Users\John\\"`** represents text value *C:\Users\John*.

Text Snippets (""")
Sometimes you need a long text value that contains some words and symbols, but also the value of a certain expression or a variable. One way to create it is to use the **`CONCAT`** function and join all the pieces of text together. A more convenient way could be writing a text snippet.

A text snippet starts with triple double quotes (**`"""`**) and continues until the next triple double quotes. It includes ends of line, so you can create multiline text values with a text snippet.

Text snippets may contain special sequences that get replaced with a calculated value:

- A dollar sign followed by a variable name is replaced by that variable's value. For example, **`"""Assigned to: $assignee"""`**.
- A dollar sign followed by an Expr expression in curly braces is replaced by the value of that expression. For example, **`"""Total score: ${ score + subtasks.score.SUM() }"""`**.



Number

A number represents a single numerical value. Aside from representing some quantity, a number value can also represent a point in time or a duration of time. In this case, you can use Format settings in the Formula column(see page 229) to properly display the results as dates or durations.

There are two forms of literal representations of numbers:

- a whole number: **`42`**
- a fractional number: **`0.239`**

Note that only dot (.) can be used as a decimal separator. Comma (,) is used to delimit function arguments. Thus, **`MAX(X, 0,618)`** will be understood as the maximum of three values: *X*, 0, and 618.

Group separators are not supported, so **`100 000`** is not a literal representation of number 100000. However, you can write a string value and explicitly call the conversion function – **`NUMBER("100 000")`**. See number conversion(see page 322).

Technical note: internally, numbers are represented as decimal floating-point numbers with 16 digits of precision and half-even rounding. Most of the operations are carried out in this form; however, some of the more sophisticated functions, such as `SQRT`, might first convert the numbers into binary floating-point, calculate the result and then convert it back into decimal floating-point.

Date and Date/Time
A Date/Time value is represented as an integer number of milliseconds since Unix epoch (1970-01-01 00:00:00 GMT). Date functions will help you convert those numbers to readable Text values.

A Date value (without the time component) is represented in the same way as the date/time value, calculated for the midnight of the specified day – *in the user's current time zone.* ⚠️ This means that expressions involving Date values (for example, using Jira's Due Date field) may produce different results for different users if they are in different time zones.

Duration
A Duration value is represented as an integer number of milliseconds. Duration functions help transform duration values into readable Text values.

Boolean
A boolean value, such as returned from comparison functions, is represented by a number `1` if true, or `0` if false.

Additionally, all values can be converted to a boolean value based on whether they are "truthy". (See Value Conversions.)

Item

An item value represents an object that you can potentially have as a row in a structure – a Jira issue, project, sprint, version, user, status, priority and others. It can also represent purely Structure-owned objects like folders.

Items have properties that you can access by writing **`item.propertyName`**, or **`(item-resulting expression).propertyName`**. The types of items, their properties and their type is listed in Item Property Reference(see page 406).

Special conversion of Item values
Item values always can be used as Text. Each item will then be represented by a text – an issue by issue key, a project by its name, etc.

Also, items may be compared using `UMAX` and `UMIN` functions, which will respect the "natural" order for the items. Most items will be compared alphabetically, but some will have a predefined order, like Priority items.

Special treatment of Item values by generators
Similar to max/min operations, if a formula column produces Item values, then you can sort by that column (or use Attribute Sorter) to reorder the structure according to the natural order of the resulting items.

If an Attribute Grouper is configured with a formula that produces Item values, the groups will become those items.

Array

An array value is a list of other values. Each value in an array is called an element of the array.

Each value in an array may be of any type – you can have an array of texts, array of items, array of arrays, and so on. An array can contain different types of values in each element – the first one could be a number, the second one an item, and the third one an array of some other elements.

The values in an array may be accessed by their index. The first value has index **0**, the second value has index **1**, and so on.

There's a number of array functions that allow you to write formulas that deal with multiple values – see Array Functions(see page 345).

Creating an array
You can create an array by calling the **ARRAY** function: **ARRAY(element1, element2, element3, ...)**

When you need to change an array, you apply some transformations to an existing array – most frequent being **FILTER** and **MAP**. This effectively creates a new array.

Accessing array elements
You can access an element of an array by its index, using the **GET** function: **array.GET(index)**

However, it's typically not needed, as the calculations with arrays are done mostly with **FILTER**, **MAP** and **REDUCE**.

Special treatment of Array values by the Attribute Grouper generator
When an array value is returned by a formula to the Attribute Grouper generator, it will create multiple groups – one for each non-empty, distinct value returned by the formula.

User Function

A user function value (sometimes called a "lambda value") represents a piece of Expr code that defines a function – see User Functions below.

Typically, you create a user function value to apply it to an array via one of the system functions, such as **FILTER** or **MAP**.

For example, consider the following code:

```
worklogs.FILTER($.author = ME()).MAP(w -> w.timeSpent).REDUCE((a, b) -> a + b)
```

There are three user functions here, **$.author = ME()**, **w -> w.timeSpent**, and **(a, b) -> a + b**. The first defines the condition for filtering work logs, the second defines how to transform each work log into a number, and the third one defines how to aggregate multiple numbers into one.

Each of these three pieces of Expr are evaluated and are represented by a value of the "User Function" type, and passed to FILTER, MAP, and similar functions.

As with any other value, a User Function value can be assigned to a variable. A more traditional function definition form could be used for that. The following two definitions are identical:

```
WITH square = x -> x * x :
...
WITH square(x) = x * x :
...
```

Key-Value Map

A key-value map is produced by some of the Expr functions, and is a collection of pairs, where "key" is the property name and "value" is the property value.

Currently, only **GROUP** function creates a key-value map, with key `group` mapping to the value the array is grouped by, and key `elements` mapping to an array of the matching elements.

To access the value, you need to write **`kvm.keyName`**, where **`kvm`** is a local variable or an expression that represents the key-value map.

Keys are supposed to be well-known; there's no way to iterate over all keys.

There's no way for the user to create an arbitrary Key-Value Map; this type is intended only for some system functions.

Errors

Errors are special values, which indicate that the calculation of some part or the whole of the expression encountered a problem.

The list of possible errors is available at Expr Error Codes(see page 414).

Normally if an error occurs somewhere in the formula, the result of the whole calculation is an error as well. But, if receiving an error from a part of the expression is legitimate, you can use the **`IFERR`** and **`ISERR`** functions to handle it.

Expr Advanced Reference - Value Conversions

Each Expr function expects a value of a certain type to be passed as each of its parameters. (See Expr Function Reference(see page 345).) When a value of a different type is encountered, an automatic conversion is attempted. If conversion is not possible, an error will result.

The conversion rules follow some basic principles:

- Best effort is made to convert simple types, such as converting a text to a number
- An item can be converted to text that represents it
- Any value can be converted to an array of one element (array "wrapped around" a value)
- An array with just one value can be converted to that value ("unwrapping" the array)

There are certain more specialized cases related to Text/Joined parameters, parameters marked as /Each and parameters of the User Function type.

The table below summarizes all the conversion rules.

Actual type → **Required type ↓**	**Number**	**Text**	**Item**	**Array**	**Key-Value Map**	**undefined**

Number	pass as is	• empty string → undefined • try to convert to a number • if unsuccessful: error	use the item's text representation, according to the item type, and then use the rules for converting Text to Number	• empty array → undefined • has only one element → try to convert that element • otherwise: error	error	undefined
Integer	• use if an integer value • otherwise: error	*same as above, and try to convert to integer*				
Date	• same as above: timestamps are represented as "unix epoch milliseconds" • dates are represented as a timestamp of the corresponding day's midnight in the user's time zone • using non-integer values may result in error					
Boolean	• false if zero • true other wise	• false if empty or only whitespaces • true otherwise (including "0"!)	true	• false if empty • true otherwise (including an array with 0 as an element)	true	false
Text	convert to text	pass as is	use the item's text representation, according to the item type	• empty array → undefined / empty text • has only one element → try to convert that element • otherwise: error	error	undefined / empty text

Text/ Joined	*same as above*			• collect a **Text** (not a Text/Joined!) value for each non-undefined element • join elements into one string with ", " as a separator	*same as above*	
Array	array with one element	array with one element	array with one element	pass as is	array with one element	undefined / empty array
Item	error	error	pass as is	• empty array → undefined • has only one element → try to convert that element • otherwise: error	error	undefined
Key-Value Map	error	error	error	• empty array → undefined • has only one element → try to convert that element • otherwise: error	pass as is	undefined
Any	any value works					

Text to Number Conversion

Some functions expect their arguments to be number values. In case an argument is a text value, we try to interpret it as a number. This can be useful if the value comes from a variable that represents a text custom field, which contains numbers — e.g., imported from some external system.

If conversion is successful, that number is used as the value for that argument. If conversion is not successful, functions can either produce an error, ignore that argument, or substitute some default — it depends on the function; see Expr Function Reference(see page 345) for details.

The first step is to accommodate for variations in number formatting. Conversion supports these formatting symbols:

- Decimal fraction separators: comma (`,`), dot (`.`)
- Digit group separators: comma (`,`), dot (`.`), apostrophe (`'`), space (`␣`)

Conversion expects that the text contains 0 or 1 decimal mark, and 0 or more group separators of the same kind. If the text contains any other formatting symbols, conversion fails. Decimal mark must come after all group separators, otherwise conversion fails.

If the text contains only one formatting symbol, and it's a dot (`.`), it is always treated as a decimal mark. If the text contains only one formatting symbol, and it's a comma (`,`), then it is treated as a decimal mark if a comma is used as a decimal separator mark in the Jira default language[53]; otherwise, it is treated as a group separator. For

53 https://confluence.atlassian.com/adminjiraserver071/choosing-a-default-language-802592304.html

instance, if the default Jira language is English, "**`101,112`**" will become 101112, whereas if it is German locale, it will be 101.112. And regardless of language, "**`1 100,23`**" will become 1100.23: space is interpreted as a group separator, and comma can only be the decimal fraction separator here.

If the group separator is a dot (`.`), then all groups except the first one must have 3 digits; otherwise, conversion fails.

After determining decimal mark and group separator symbols, conversion removes all group separator symbols and replaces the decimal mark with a dot. Note that if text contains several whole numbers separated by spaces, conversion will think it is one number, for example, "**`10 11 12`**" will become 101112. Similarly, "**`10,11,12`**" will become 101112.

The final step of conversion is to recognize the resulting text as either Expr's literal number representation or scientific or engineering notation. Examples:
`0.239`
`-1.32e5`
`12e-3`

Conversion to Boolean: Falsy and Truthy Values

A value is *falsy* if it is:

- `undefined`,
- number 0,
- an empty text value (`""` or `''`), or a text value that contains only space characters,
- an empty array.

All other values are *truthy.*

When converting to a Boolean, truthy values become true and falsy values become false.

By convention, when functions or logical operators need to construct a truthy value, they use the number `1`.

Text vs. Text/Joined

When a function declares that it requires "Text/Joined" value, it means that the value will be converted to a text, with an additional special handling of the array type:

- If it's an empty array, or an array with only one element, the conversion will be the same as for "Text" type.
- If an array with multiple values is passed, then a) each element of the array will be converted to a Text value, b) all these texts will be joined together with a comma as a separator.

Here's an example illustrating the difference when fixVersion is passed as a parameter - notice that in the third row, there are multiple values in fixVersion (because it's an array), so Text and Text/Joined are treated differently:

fixVersion	**Function accepting Text will receive**	**Function accepting Text/Joined will receive**
(no value)	undefined	undefined
v1	"v1"	"v1"
v1, v2	error	"v1, v2"

Passing Implicit User Function as a Parameter

You can always pass an implicit User Function (the one containing the "$" symbol) as an argument to a function. This will result in the call to this function to become an implicit User Function value itself.

For example, consider the following expression:

```
fixVersions.FILTER(YEAR($.releaseDate) = 2021)
```

Function **`YEAR()`** expects a date, but it receives a User Function instead (**`$.releaseDate`**), which produces the release date for each passed version. As a result of applying **YEAR()** to that User Function, we will get another user function, which produces the *year of the release date* for each passed version.

This logic applies only to the implicit User Functions, defined with the $ sign.

The functions that expect a User Function parameter, like FILTER, are exclusions from this rule.

Variables

Variables (also known as free, or externally set variables) represent some values that will be fed into the formula for each Structure row that the formula will be applied to.

For example:

```
IF priority = "Blocker" : parent.estimate + x
```

In this formula, "priority", "parent" and "x" are all variables – they vary from one row to another. (They will not change the value while calculating the expression for a single row.)

There's no need to declare a variable, you can immediately start using it. All valid identifiers that are used like variables will be treated as such.

Each formula is expected to contain at least one variable – otherwise, the result will be the same for each row.

Variable to Attribute Mapping

Each variable should be mapped to a valid attribute – such as a Jira field or a Structure attribute, so when an expression is calculated for a particular item, the value of that attribute becomes the variable's value.

If you use one of the well-defined variable names, it will be automatically mapped to the corresponding attribute. See Standard Variable Reference(see page 402).

If you use an arbitrary variable name, such as "x", you will need to map it as described on the Mapping Variables(see page 232) page.

If a variable is not mapped, or if the item does not support the mapped attribute, the value of the variable will be the `undefined` value.

"this" Variable

One of the well-defined variable names is "this". It is mapped to an attribute that provides a value of Item type, representing the item for which the formula is being calculated.

This may come in handy in certain cases. For example, to analyze issue links and pick the "other side" of a link, regardless of whether it's an incoming or an outgoing link:

- **`issueLinks.MAP(IF $.source = this : $.destination ELSE $.source)`**

Alternatively, you can use "item" with the same meaning.

Expr Advanced Reference - Local Variables

Local variables are similar to Variables, but they are not mapped to an item's attribute or Jira field, but rather defined and calculated right in the expression.

The declaration syntax is the following:

Note the colon ("**:**") that separates the expression assigned to the variable and the expression where the variable is used.

A few facts about local variables:

- *ExpressionWithLocalVariable* may start with another local variable definition, so you can introduce many local variables in sequence. When defining a second variable, you can use the first variable already defined, and so on.
- Local variables can "shadow" previously defined local and free (mapped) variables with the same name. If you write **`WITH priority = 10: <expression>`**, then when calculating `<expression>`, the value of **`priority`** will be 10, even if there was a variable attached to the issue's priority in the enclosing scope.
- The **`WITH...`** construct is itself an expression, so you can use it, enclosed in parentheses, anywhere an expression can be used. The name defined in this expression is not visible outside the **`WITH...`** expression.

Immutability

Expr language constructs are immutable. Once a local variable is defined, it cannot change its value. (So, in fact, calling it a "variable" is not exactly correct. Although, if a local variable depends on external variables, which vary from item to item, the local variable itself will also vary from item to item.)

So if you're building your formula and you need to take a number of values through a series of calculations, you may need to use multiple local variables, going through each step and assigning each intermediate result to another local variable.

Expr Advanced Reference - Operators

Operators allow writing formulas in a convenient way using traditional logical and arithmetic operations. Each operator has one (unary operators) or two operands. Each operand could be a variable, a number, or just any expression, which sometimes will need to be in parentheses.

The operators translate into calling the corresponding functions on operands as arguments. If an operand expression evaluates to an error, then the operator's result will also be an error.

Available operators are:

Operator(s)	Symbol(s)	Priority	Type of Operands	Result Type	Corresponding Function(s)
Logical negation	`NOT`	7	Any	Number (Boolean)	`NOT`
Unary sign	`+ -`	7	Convert to Number	Number	`SUM, MINUS`

Operator(s)	Symbol(s)	Priority	Type of Operands	Result Type	Corresponding Function(s)
Multiplication and division	`* /`	6	Convert to Number	Number	`MUL, DIV`
Addition and subtraction	`+ –`	5	Convert to Number	Number	`SUM, MINUS`
Concatenation	`CONCAT`	4	Convert to Text/ Joined	Text	`CONCAT`
Equality/Inequality check	`= !=`	3	Any	Number (Boolean)	`EQ, NE`
Numeric Comparison	`< > <= >=`	3	Number	Number (Boolean)	`LT, GT, LE, GE`
Logical AND	`AND`	2	Any	Any(*)	`AND`
Logical OR	`OR`	1 (lowest)	Any	Any(*)	`OR`

The operators are listed in their priority order. The priority is important when an expression could allow different interpretations about which operators are applied first.

For example, in an expression **`progress + parent.progress * weight < threshold AND priority != "Blocker"`**, the multiplication is executed first, then the addition, then the comparisons, and then the logical AND. If you'd like to alter the order of operator application, use the parentheses.

Logical Negation (NOT)

To negate a logical value or expression, use the **`NOT`** operator. Instead of **`NOT`**, an exclamation mark (**`!`**) can also be used.

The operator produces **`0`** if the operand is a truthy value, and **`1`** otherwise. Therefore, it may be applied to any value.

If the value negated is an expression with other operators, it should be contained in parentheses.

Examples:

- **`NOT resolved`**
- **`NOT (storyPoints > 0 AND storyPoints < parent.maxStoryPoints)`**

Unary Sign (+ -)

The operator first attempts to convert the value of an expression to a number. If conversion succeeds, + produces this number, and – produces the negated number.

If the conversion to a number fails, and the value of the expression is falsy, the negation produces `undefined`. Otherwise, it produces an error.

Arithmetic operators (* / + -)

Arithmetic operators are: addition (+), subtraction (-), multiplication (*) and division (/).

Multiplication and division have precedence over addition and subtraction.

These operators convert their arguments to numbers. A non-empty, non-number argument would produce an error. Falsy non-number values are treated as zero.

Examples:

- `"" + 1 → 1`
- `"foo" + 1 → error`
- `"" * 1 → 0`
- `"foo" * 1 → error`
- `"" - 1 → -1`
- `1/0 → error`

If any subexpression produces an error, the operator produces the same error.

Concatenation (CONCAT)

The concatenation operator converts each operand to a text value (Text/Joined, to be precise) and creates a new text by joining them together. It is identical to calling the CONCAT function on its operands:

```
value1 CONCAT value2 CONCAT value3 = CONCAT(value1, value2, value3)
```

The operator can be used to increase the readability of a formula.

Note that CONCAT is applied after all the arithmetic operators but before comparisons and logical operators.

Equality check (= !=)

The "equals" operator (=) checks that both arguments are "essentially the same value". The "not equals" operator (`!=` or `<>`) produces the inverse value.

The comparison rules are based on the type of the values compared.

Equality operator will return true (1) if any of the following conditions hold:

- Both values are **Undefined.**
- One value is a **Number**, and the second value is the same number, or can be converted to the same number.
- One value is a **Text**, and the second value can be converted to Text (using Text/Joined), and both values are "essentially the same".
    - The differences in letter forms and leading and trailing whitespace are ignored (thus `"  cote  " = "côte"`).
- One value is an **Item**, and the other value is the same item. (Note that if the other value is a Text, both values can be compared as text values.)
- One value is a **Key-Value Map**, and the other value is also a key-value map with the same contents (these equality rules applied to all elements).
- One value is a **User Function**, and the other value is exactly the same user function (not just having the same logic).
- One value is an **Array**, and either of the following is true:
    - The other value is also an **array**, has the same number of elements and the elements are respectively equal.
    - The other value is **undefined** and the array does not contain any non-undefined elements.
    - This array contains only **one element** and this element is equal to the other non-array value we are comparing the array with.

In all other cases, the equality returns false (`0`).

Note that you can compare an item to a text, for example,

```
IF project = "My Project" : ...
```

The item will get converted to a text using value conversion rules described earlier.

If one value is a number and the other value can be converted to a number, both values are treated as numbers. However, if both values are text, they will be treated as text, even if both can be converted to a number. You can use the NUMBER(see page 0) function to force a value to be numeric.

- `3.4 = 3.40 → 1`
- `3.4 = "3.40" → 1`
- `"3.4" = "3.40" → 0`
- `NUMBER("3.4") = "3.40" → 1`

Numeric Comparison (< > <= >=)

The ordering / comparison operators work on numbers only:

- **<** (less than)
- **>** (greater than)
- **<=** (less than or equal)
- **>=** (greater than or equal)

If either of the values is text, the operator attempts to convert it to a number. If the conversion fails, the result is an error.

If any value is undefined, strict operators (**<**, **>**) produce `0`. Non-strict (**<=**, **>=**) produce `0`, unless *both* values are undefined (because they are equal).

Logical operators (AND OR)

The logical operators are used to combine other logical conditions, or to pick an alternative or a conditional value.

- **OR** also can be written as "**||**" or "**|**"
- **AND** also can be written as "**&&**" or "**&**"

When both operands are Number(Boolean), then the operators perform the corresponding boolean operation.

However, you can use these operators with non-boolean operands in a "short circuit" way, based on whether the operands are "truthy" (see above).

- **`a OR b`** – if "a" is truthy, "b" is not evaluated and the result of the operation is "a"; otherwise the result of the operation is "b".
- **`a AND b`** – if "a" is falsy, "b" is not evaluated and the result of the operation is "a"; otherwise the result of the operation is "b".

Note that "b" is not evaluated in case the result of the operation is equal to "a". This might be important in case calculating "b" could result in an error.

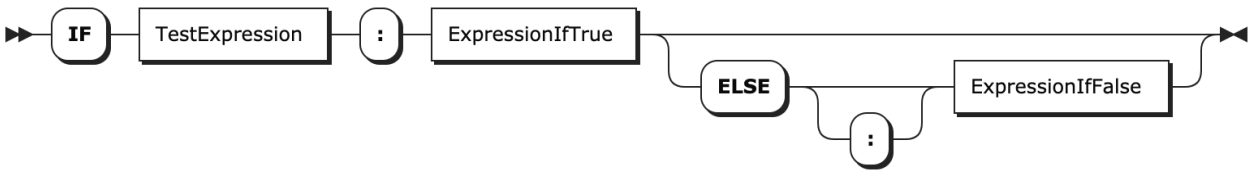
Examples:

- **`assignee OR "UNASSIGNED"`** – This will produce either the issue's assignee user key or (if the issue is unassigned) the text value "**`UNDEFINED`**".
- **`!assignee AND status = "OPEN"`** – This will produce **`1`** if the issue is unassigned and in status **`OPEN`**, and **`0`** otherwise.

- **`count AND total / count`** – This will produce some average number (total / count), unless count is 0 – in this case the result will be 0. Note that there will be no division by zero error.

Expr Advanced Reference - Conditional Expressions

A conditional "IF" expression allows you to switch between two expressions, based on whether a condition is true (truthy) or false. It is identical to calling the IF function with two or three arguments.



The "ELSE" part, as well as the colon ("**:**") after ELSE are optional. If the ELSE part is omitted, and the test expression evaluates to false, the result is `undefined`.

If you use nested IF expressions with only one ELSE, the ELSE part applies to the innermost IF. We recommend using parentheses to make it clear which IF it applies to.

You can use indentation to make the formula with nested IFs more readable – but the indentation has no effect on how the formula is parsed. Use parentheses!

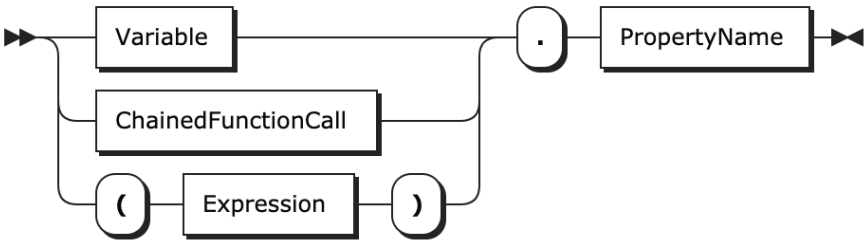
Examples:

```
IF assignee = ME() : "mine!"
---
IF dueDate < NOW() : "overdue!" ELSE: """${DAYS_BETWEEN(NOW(), dueDate) - 1} days
left!"""
---
IF priority = "Critical":
IF dueDate < DATE_ADD(TODAY(), 7, "days"):
   "critical and urgent"
ELSE:
   "critical and not urgent"
```

Expr Advanced Reference - Property Access

Some Expr values – Items and Key-Value Maps – contain "properties", named values that are parts of the bigger value. For example, a Version item contains properties such as "name", "description", "releaseDate". And function GROUP returns a value that is an array of Key-Value Maps, each map having "group" and "elements" properties.

To access a property, use the dot ("**.**"), followed by the property name. The names are case-insensitive.



Examples:

- **`version.releaseDate`**
- **`sprint.startDate`**
- **`issueType.isSubtask`**

An alternative way to access a property is by using the **`ACCESS()`** function – in this way, the property name itself may be calculated:

- **`parentTask.ACCESS("Story Points")`**

⚠ If the value does not have the requested property, or if the value is neither an Item nor a Key-Value Map, the resulting value is `undefined` – not an error! (See also a note about arrays below.)

ⓘ There's no way to check if a value contains a certain property (other than try and access it), or list all available properties for an item.

Accessing Custom Fields via Properties

In most cases a formula will refer to the issue's custom fields directly by name – for example, the formula **`impact / storyPoints`** uses the values of two custom fields, Impact and Story Points, of the currently calculated issue to calculate the benefit-to-cost ratio.

You can, however, use a formula to access other, related issues, and their corresponding custom fields – for example, the formula **`impact / (storyPoints + subtasks.storyPoints.SUM())`** calculates a similar value but takes into account the cost (in story points) of all the subtasks.

Note that the actual custom field is named "Story Points", with a whitespace. When you access a property of an issue item, Structure tries to match the property name with available custom field names using loose rules, dropping whitespace and any non-identifier-friendly characters. The property name is also case insensitive.

You can also use the ACCESS() function to specify the name of the field precisely, or use "customfield_NNNNN" property name to identify a field by its ID.

- **`parent.storyPoints`**
- **`parent.ACCESS("Story Points")`**
- **`parent.customfield_10000`**

✓ You can use `this` variable to access properties of the currently calculated item. Normally it's not needed, since the same values are available as corresponding variables.

Accessing Property of Each Element in an Array

You can apply property access to an array of values. Expr will then apply property access to each element and return the result as an array.

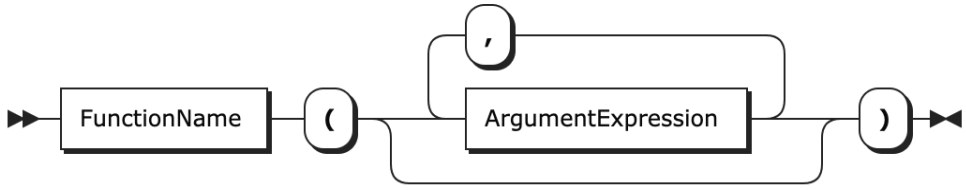
For example:

- **fixVersion.releaseDate** – will return an array of release dates
- **worklogs.author.UNIQUE()** – will return a list of people who logged work
- **subtasks.remainingEstimate.SUM()** – will return a total remaining estimate from the subtasks

In the resulting array, all **`undefined`** results will be removed and the array will be flattened. This is the same behavior as shown by functions with /Each parameter type.

Expr Advanced Reference - Functions

A function takes zero or more values, and produce another value. A function call consists of a function name (an identifier), followed by its arguments enclosed in parentheses. An argument can be any expression. Different arguments are separated by commas (**`,`**) or semicolons (**`;`**) — for one function call, all separators must be the same.



Examples:

- **`NOW()`**
- **`ROUND(storyPoints / 10)`**
- **`FILTER(sprint, $.state = "active")`**
- **`MAKE_DATETIME(2017; 12; 31; 23; 59; 59)`**
- **`myUserFunction("argument1", "argument2")`**

A function call can evaluate only some or even none of the arguments, depending on the function. This is useful for functions that perform choices. For example, in an IF function, the argument that wasn't chosen is not evaluated, so the whole expression doesn't produce an error when that argument produces an error.

System and User Functions

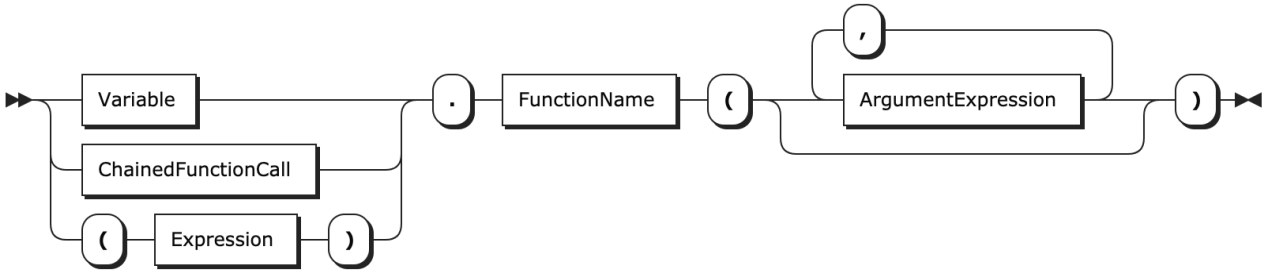
System functions are provided by Structure. The functions are listed in the Expr Function Reference(see page 345). Each function expects a certain number and type of parameters.

User functions are explained in the next section. A user function is called using the same syntax as a system function.

> User functions can take any number of arguments, regardless of how many arguments are declared. If a parameter was declared, but a value was not provided when calling the function, the parameter's value will be `undefined`.

Chained Function Calls

A different way to call a function is by "chaining" it to its first argument by adding a period ("**.**"), a function name, parentheses, and any additional parameters, if any.



For example, **`sprint.FILTER($.state = "active")`** is the same as **`FILTER(sprint, $.state = "active")`**.

This allows nice, readable expressions, where a value is sequentially transformed by applying functions to the result of a previous function call:

- affectsVersion.MAP($.releaseDate - $.startDate).MAX()
- linkedIssues.FILTER(x -> NOT x.resolution).MAP(x -> x.remainingEstimate).SUM()

Unlike some other languages, in Expr any function may be written in the chained syntax, regardless of what the value is.

Applying Functions to Arrays

When a function is applied to an array value (meaning that an array is passed as the first argument of the function), the result may be calculated in a number of ways, depending on which function is called and what type of argument it expects.

You can always apply a function to each element of an array using the MAP function. For example:

- **`array.MAP(SOME_FUNCTION($))`**

The cases below relate to cases where an array is passed directly as an argument: **`SOME_FUNCTION(array)`**.

Using an Array Function

There are special functions that expect an array as their first argument – **`FILTER`**, **`MAP`**, and others(see page 337).

There's no special behavior in this case – an array is expected. In fact, if the value passed is not an array, it will be converted to either an array of one element (containing that value), or an empty array if the value is undefined.

Passing Array as an Argument of Text/Joined Type

If a function declares that it expects "Text/Joined" as an argument, then the system will try to convert an array into a text value. See the "Text vs Text/Joined" section above.

For example:

- **`CONCAT("Versions: ", fixVersion)`** → **`"Versions: v1, v2, v3"`**

Passing Array as an Argument of /Each Type

If a function declares that it expects "Number/Each" or "Text/Each" or any other "/Each" type as an argument, then it would work on that simple type, but if an array is passed, it will apply its logic to each element in that array. The result of calling this function will be an array, where each element is a result of applying the function to the original element.

For example:

- **`UPPER(fixVersion) → ARRAY("V1", "V2", "V3")`**

In addition, when applying a function to an array in this way, the resulting array is "flattened" (elements from any sub-arrays moved to be the elements of the top array) and "compacted" (all undefined elements are removed).

Passing Array to a User Function
You can call a user function and pass an array as an argument. No special handling takes place – just as with a system function expecting an array.

All Other Cases
If a system function does not expect an array, but it is passed as an argument, it will try to convert it to the value type it expects. A one-element array will be converted to its single element and an empty array will be converted to `undefined`. See "Value Conversions" above.

If the conversion is not possible, the result will be an error.

Expr Advanced Reference - User Functions

User Functions are functional *expressions*, defined by the user. (They could also be called "lambdas".) User functions are helpful when the user needs to apply some repetitive action, or to pass an action to be applied to each item in an array.

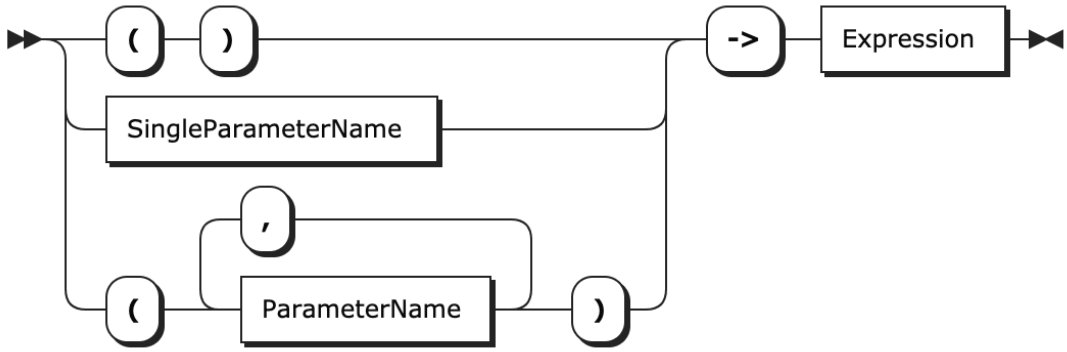
A User Function contains a list of parameters and then an expression that is calculated for these parameters.

A User Function is a *type of value*, so you can assign a user function to a local variable, or pass it to some higher-order function as a parameter.

There are three ways to define a user function.

Functional Expression

A functional expression is the canonical form for user functions. It contains a list of parameters in parentheses, followed by the "maps to" symbol (**->**), followed by the expression calculated by the function. When there's only one parameter, the parentheses can be omitted.



Examples:

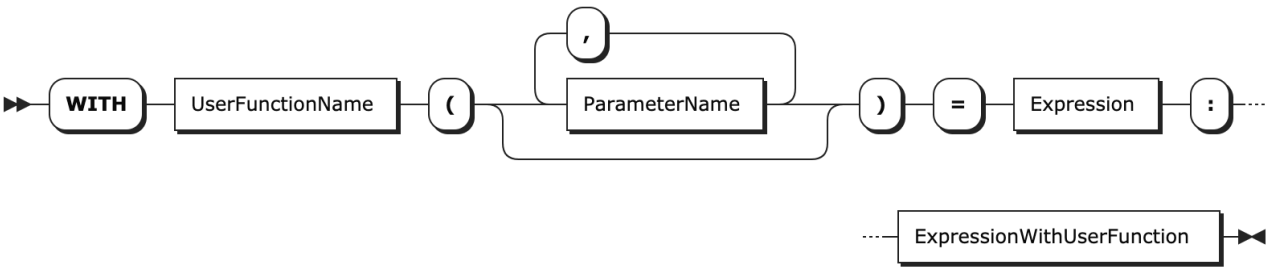
- `() -> START_OF_MONTH(NOW())`
- `(x) -> x * x`
- `version -> version.releaseDate - version.startDate`
- `(s1, s2) -> s1 CONCAT " " CONCAT s2`

All these examples evaluate to a "User Function" value type, which can be assigned to a variable:

- `WITH square = x -> x * x : ...`

Traditional Function Definition

A more traditional function definition looks similar to a variable definition, only the variable is followed by a list of parameters in parentheses, and the expression is based on those parameters.



To rewrite the examples above:

- `WITH currentMonth() = START_OF_MONTH(NOW()) : ...`
- `WITH square(x) = x * x : ...`
- `WITH versionDuration(version) = version.releaseDate - version.startDate : ...`
- `WITH joined(s1, s2) = s1 CONCAT " " CONCAT s2 : ...`

These declarations are identical to the corresponding examples in the previous section, with local variables assigned to the corresponding User Function values.

Implicit Functional Expression ($)

Most of the time when we're creating a formula with an array, we need to apply some kind of operation to each element of the array. Implicit functional expressions help define the corresponding user function easily by having "$" denote "each element".

For example:

- `versions.FILTER($.startDate < NOW())`

- issueLinks.FILTER($.type = "Relates").MAP($.destination)
- **`worklogs.UMAX_BY(IF $.author = ME() : $.timeSpent)`**

In each case, the expression with "$" is transformed into a User Function with a single parameter, which is then substituted for $. So, the last example from the list above is identical to:

- **`worklogs.UMAX_BY(w -> IF w.author = ME() : w.timeSpent)`**

When reading these expressions, you can say "each" when the dollar sign is encountered.

An implicit user function must always be used in an argument to a system function, which expects a user function. Otherwise, it won't be accepted.

For example – here's how we can filter an array to contain only even numbers:

Correct	Incorrect – Parse Error

`ARRAY(1, 2, 3).FILTER(MOD($, 2) = 0)`
... or, alternatively ...
`WITH even(e) = MOD(e, 2) = 0 :`
`ARRAY(1, 2, 3).FILTER(even)`
	`WITH even = MOD($, 2) = 0 :`
	`ARRAY(1, 2, 3).FILTER(even)`

Calling User Functions

If a User Function is assigned to a variable, you can call it in your expression in the same way you call a system function.

- **`WITH square(x) = x * x :`**
  `square(impact) / square(cost)`

You can also use the chained function call notation:

- **`WITH square(x) = x * x :`**
  `WITH fquare(x) = x.square().square() :`
  `storyPoints.fquare()`

Note that you cannot invoke a functional expression unless it is assigned to a local variable. The following will produce an error: **`(x -> x * x)(3)`**

Function Name Collisions

Both system functions and user functions are invoked in the same way – **`FUNCTION_NAME(arg1, arg2, arg3, ...)`**, or with a chained call syntax – **`arg1.FUNCTION_NAME(arg2, arg3, ...)`**. This leaves a potential for the user to define a function that has the same name as a system function.

⚠️ When Expr encounters a function call, first it looks up if there is a *system* function of that name. ⚠️

The system function will be called even if there's a local variable of the same name. To protect the user from name collisions, Expr will show an error if you try to define a function with a name that matches a system function name. (However, it won't be able to detect the collision if a local variable is defined through a series of assignments of a functional expression.)

You *can* define a local variable of any other type with a name identical to a system function's name.

Works as expected	Error

`// Using "SUM" as a local variable,`
`// but "SUM()" is also a system function`
`// and "SUM{}" is an aggregate function.`
`WITH SUM = cost + parent.cost :`
`SUM(SUM, SUM { cost })`
	`// Cannot define a user function with a name collision.`
	`// Note that the language is case-insensitive: "SUM" and "sum" are the same.`
	`WITH sum(issue) =`
	`  issue.timeSpent + issue.parent.timeSpent :`
	`...`

Note that the function name collision resolution provides potential challenges when upgrading to a newer version of Structure, if that version introduces new system functions.

Let's say you have defined a user function **`LAST_COMMENT()`** in your formula and used it successfully in an older Structure version. If the newer version of Structure adds a system function **`LAST_COMMENT()`**, that formula will likely stop working after the upgrade, and you will need to rename the user function.

To minimize the probability of this happening, we suggest naming your user function in a way that makes potential collision unlikely. It could be a name that is very specific to your configuration, or you can always prepend the name with an underscore – in our example, call it **`_LAST_COMMENT()`**.

Expr Advanced Reference - Aggregate Function

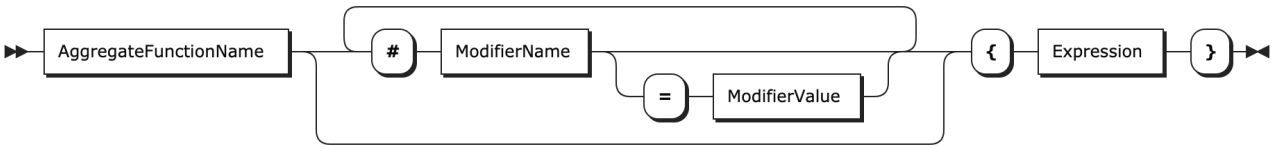
An aggregate function calculates values for some other rows in the structure (for example, for all sub-items), aggregates these values (for example, adds them together), and produces the resulting value to be used in the formula.

For example:

- **`SUM#children { storyPoints }`** – calculates total story points for all the child issues.
- **`PARENT { fixVersion }`** – provides the value of the fixVersion field from the parent's issue.
- **`VALUES { components }`** – collects all distinct components that are set for any of the sub-issues of the current row.

Aggregate functions allow you to calculate complex values that depend not only on the "current" item, but also on other items in relation to it.

An aggregate function starts with a name, optionally followed by modifiers, then curly braces ("**`{}`**"), and inside them – an "inner expression", which will be calculated for some other rows. You can use whitespace between any elements of the aggregate function calls.



The inner expression may return any type of value – number, text, array, and others – *except a user function*. You cannot pass a user function from the inner formula into the outer formula.

Available Aggregate Functions are listed in Aggregate Function Reference(see page 387).

Aggregate Function Modifiers

An aggregate function may have one or more modifiers that govern the aspects of the function's execution. Each modifier starts with hash sign ("`#`"); then comes the modifier's name; optionally followed by the equals sign ("`=`") and a value, which can be a string or a numeric constant. If a value is omitted, it is assumed to be `1` (a representation of *true* in Expr).

Examples:

- **`SUM#all { cost }`**
- **`SUM#all#leaves { IF type = "story": storyPoints }`**
- **`JOIN#separator=", " { key }`**
- **`JOIN #separator=", " #fromDepth=0 #toDepth=-1 { key }`**

Each aggregate function supports a specific set of modifiers, not all of them. Using an incompatible modifier will result in an error. To learn more about available modifiers and their restrictions, see Aggregation Modifiers(see page 393).

Sharing Values Between Outer and Inner Formulas

It's important to understand that the formula inside an aggregate function – the inner formula – is calculated fully separately from the "outer" formula. Both formulas will share variable mappings (to attributes), but any local variables and user functions defined on one side will not be accessible from the other side.

Will not work!

```
1  // Cannot use "total" on line 4!
2
3  WITH total = SUM#children { storyPoints } :
4  WITH median = MEDIAN#children { storyPoints / total } :
5  ...
```

Correct version

```
1  WITH median = MEDIAN#children {
2    WITH total = PARENT { SUM#children { storyPoints } } :
3    storyPoints / total
4  } :
5  ...
```

As you can see from the example above, you may need to use nested aggregate functions instead.

Using Formulas with Aggregate Functions in Generators and Transformations

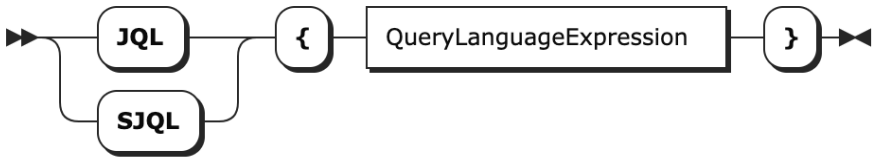
Note that when you use an aggregate function, it relies on the existing structure to figure out the parent item, child items, and other related items addressed by the formula. When the formula is used to build or transform a structure, the hierarchy and items that the aggregate functions "see" will correspond to the structure that exists immediately before the generator or the transformation is applied.

Therefore, when using a formula with Attribute Filter, Attribute Grouper, Attribute Sorter and other generators, you should apply aggregate functions carefully, understanding what is the preceding structure before the generator is

applied. For example, if you use a grouper on a folder with a flat list of issues, the formula in the grouper will see issues and the folder will be their parent item – there are no groups yet!

Expr Advanced Reference - Embedded Queries

You can embed JQL and S-JQL(see page 428) in a formula to check if the item (the one the formula is being calculated for) satisfies the condition of the query – that is, it will be a part of the query result.



The syntax is similar to calling an aggregate function:

- `IF JQL { assignee in membersOf("Team-Alpha") } : ...`
- `IF NOT SJQL { descendant of folder("Excluded") } : ...`

The result of evaluating `JQL {}` or `SJQL {}` is always `0` (false) or `1` (true).

When SJQL is used, it is always applied to the current structure. (Or the one being generated – see the note below.)

Note that, unlike aggregate functions, these constructs do not use Expr, but rather another other languages, as the inner expression. Use the corresponding documentation as a reference for JQL and S-JQL.

> The embedded queries are calculated separately from the Expr formula they are used in. Therefore, you cannot use any values from the Expr formula inside a JQL or SJQL query or vice versa. Also, you cannot check the query match for any other item except the one the formula is being calculated for.
>
> In other words, the only data that is passed between an embedded query and the outer Expr formula is 1 or 0 depending on whether the current item matches the query.

Using S-JQL in Generators and Transformations

An S-JQL query usually depends on the structure it's being calculated for. So, similar to Aggregate Functions, when you use a formula with an embedded S-JQL in a generator or a transformation, the query will be applied to the underlying "preceding" structure, that exists before the generator is applied.

S-JQL Query Performance

Structure optimizes the calls to embedded queries. A query will be run only once for multiple items that the formula is being calculated for, and the result will be checked separately in the calculations for each row.

That said, the JQL itself may potentially be an intensive calculation, if it uses JQL functions or historical conditions like WAS. Please be careful when trying this JQL in a formula, and watch for how long the value is being calculated before publishing the formula for other users. Normally, calculating a "heavy" formula should not prevent users from doing other things in Structure (including using other columns with formulas), but it can place some stress on the Jira server.

> Please avoid using the `structure()` JQL function in JQL or S-JQL that is being embedded in a formula.

3.8.3.2 Expr Function Reference

All standard Expr functions are listed below, grouped by category. Use the right navigation panel to scroll through functions, or Ctrl+F to find a specific function.

For more information about how functions work, see Notes About Functions(see page 345).

Notes About Functions

A function may take zero, one or more arguments. Some functions can take an unlimited number of arguments.

Parameter Types

Functions expect certain value types for each parameter. When an unexpected type is provided, Structure will attempt to convert it to an acceptable type. If it can't, the function may consider the value undefined and ignore the value, or it may return an error.

For example, the MIN function is used to select the smallest number from a series; if the text value "10" is passed as of the parameters, it will convert that to the number 10. If "ABC" is passed, it won't be able to convert that to a number and will return an error.

For more information about how Structure converts values, see Value Conversions(see page 0).

Undefined Values

A variable used in a formula may have `undefined` value. Usually it means that the value for an issue is not set – for example, Resolution field will produce `undefined` value until the issue is resolved. When a function that manipulates values receives an `undefined` value as its primary argument, the return value will also typically be undefined.

Arrays

Many functions can also be applied to arrays. When the argument is an array, it will be treated in one of the following ways:

- Numbers and Text - If the function is expecting a number or text, it will attempt to convert the first element of the array into the appropriate type. If it fails, it will return an error. If there are more than one elements in the array, it will also return an error.
- Text/Joined - If an array is passed as the parameter, each element in the array will be converted to text and joined together with a comma. (See Text vs. Text/Joined(see page 305))
- /Each - If an array is passed as the parameter, the function will be applied separately to each element in the array. The result will be an array with the function applied to each value.

Array Functions

ALL

ALL(A, $)

Checks that **$** returns a truthy value for all elements in the array.

Parameter	Type	Description
A	Array	Array of elements to be used in $.
$	User Function	Function to be applied to each element.
→ Result	Boolean	Returns true if $ returns a truthy value for all elements in the array. Otherwise, returns false. If the array is empty, returns true.

Examples:

- `ALL(fixVersions, $`[54]`.startDate > TODAY())`→ Returns true if all fixVersions start after the current date.

ANY

ANY(A, $)

Checks if the array has at least one element for which **$** returns true.

Parameter	Type	Description
A	Array	Array of elements to be used in $.
$	User Function	Function to be applied to each element.
→ Result	Boolean	Returns true if $ returns a truthy value for at least one elements in the array. Otherwise, returns false. If the array is empty, returns false.

Examples:

- `ANY(fixVersions, $`[55]`.startDate < TODAY())`→ Returns true if any fixVersions started before the current date.

54 http://expr.pocker.almworks.com/secure/StructureBoard.jspa?s=3
55 http://expr.pocker.almworks.com/secure/StructureBoard.jspa?s=3

ARRAY

ARRAY(Element1, Element2, ..., ElementN)

Creates an array from a list of elements.

Parameter	Type	Description
`Element1, Element2, ..., ElementN`	`Any`	Elements to be added to the array.
→ Result	`Array`	Array containing all of the elements.

Example:

- `ARRAY(1, 2, 3)` → `ARRAY(1, 2, 3)`

COMPACT

COMPACT(A)

Removes all undefined values from the array.

Parameter	Type	Description
`A`	`Array`	Array to be compacted.
→ Result	`Array`	Compacted array.

Note: error values and empty arrays are preserved.

Examples:

- `COMPACT(ARRAY(1, 2, undefined, 3))` → `ARRAY(1, 2, 3)`

CONTAINS

CONTAINS(A, Element)

Searches an array for a specified element. The comparison is done in the same way function EQ (=) works.

Parameter	Type	Description
`A`	`Array`	Array to search in.
`Element`	`Any`	Element to look for.
→ Result	`Boolean`	Returns true if the array contains the specified element. Otherwise, returns false.

Examples:

- `CONTAINS(ARRAY(1, 2, 3), 2) → 1`
- `CONTAINS(ARRAY(1, 2, 3), 5) → 0`
- `fixVersions.CONTAINS("v1") →` returns 1 if one of the versions in the Fix Version/s field is "v1"

Please note that `CONTAINS` does not perform text matching.For example, if Fix Version/s field contains version "v1.1", the above example – `fixVersion.CONTAINS("v1")` – will return 0. You can apply text matching(see page 417) to look for elements that match a particular pattern, for example: `fixVersion.ANY($.MATCH("v1*"))`

CONTAINS_ALL

CONTAINS_ALL(A, Elements_Array)

Searches an array for all of the elements passed in the **Elements_Array** parameter.

Parameter	Type	Description
`A`	`Array`	Array to search in.
`Elements_Array`	`Array`	Array of elements to look for.
→ Result	`Boolean`	Returns true (1) if `A` contains all elements contained in `Elements)Array`. Otherwise, returns false (0).

Duplicates are not taken into account, so `CONTAINS_ALL(ARRAY(1), ARRAY(1,1)) → 1 .`

Examples:

- `CONTAINS_ALL(ARRAY(1, 2, 3), ARRAY(1, 2, 3)) → 1`
- `CONTAINS_ALL(ARRAY(1, 2, 3), ARRAY(1, 2, 4)) → 0`

CONTAINS_ANY

CONTAINS_ANY(A, Elements_Array)

Searches an array for any of the elements passed in the **Elements_Array** parameter.

Parameter	Type	Description
`A`	`Array`	Array to search in.
`Elements_Array`	`Array`	Array of elements to look for.
→ Result	`Boolean`	Returns true (1) if `A` contains any elements contained in `Elements_Array`. Otherwise, returns false (0).

Examples:

- `CONTAINS_ANY(ARRAY(1, 2, 3), ARRAY(2, 9, 7)) → 1`
- `CONTAINS_ANY(ARRAY(1, 2, 3), ARRAY(4, 9, 7)) → 0`

FILTER

FILTER(A, $)

Filters the values in an array and produces a new array that retains only elements for which the user function **$** returns a true value (considered as a **Boolean**).

Parameter	Type	Description
`A`	`Array`	Array of elements to be filtered.
`$`	`User Function`	Function to be applied to each element.
→ Result	`Array`	New filtered array.

Example:

- `ARRAY(100, 200, 300).FILTER(x -> x < 250) → ARRAY(100, 200)`
- `worklogs.FILTER($.author = ME()) →` Returns an array containing worklog values created by the current user

FIRST

FIRST(A)

Returns first element of the array, or undefined if the array is empty.

Parameter	Type	Description
`A`	`Array`	Array of elements.
→ Result	`Any`	First element contained in the array. If empty, returns undefined.

Example:

- `FIRST(ARRAY(1, 2,3)) → 1`

FLATTEN

FLATTEN(A)

Given an array of arrays, makes a single array, using one-step flattening (each element in the sub-arrays is added to the top array as a single element).

Parameter	Type	Description
A	Array	Array of arrays, or array containing arrays and non-array values.
→ Result	Array	A single array, containing all elements.

Example:

- `FLATTEN(ARRAY(ARRAY(1, 2), 100, ARRAY(2, 3), 10)) → ARRAY(1, 2, 100, 2, 3, 10)`

GET

GET(A, Index)

Retrieves an element from an array based on its index. Array indexes are 0-based. Returns undefined if index is out of array bounds.

Parameter	Type	Description
A	Array	Array to search.
Index	Integer	Numeric index, 0-based: 0 corresponds to the first value in A, 1 corresponds to the second, and so on.
→ Result	Any	The value corresponding to Index.

Example:

- `GET(ARRAY(1, 25, 2, 18, 100), 1) → 25`

GROUP

GROUP(A, $)

Groups array elements into buckets based on the value produced by the user function **$** for each element. The result is an array of groups (key-value maps).

Parameter	Type	Description
A	Array	Array of elements to be grouped.
$	User Function	Function to be applied to each element.

Parameter	Type	Description
→ Result	`Array`	Array of groups. Each group G has **G.group** which contains the value that the user function produced, and **G.elements**, which contains an array of all elements that produced that value. The order of groups corresponds to the order of the grouping values as they first appear in the source array.

Example:

Suppose the ***worklogs*** attribute contains the following work logs for the issue:

- Author: Bob, Time Spent: 1 hour, Date: Feb-1
- Author: Alice, Time Spent: 2 hours, Date: Feb-1
- Author: Bob, Time Spent: 3 hours, Date: Feb-2

Let's write this using the following pseudo-formula:

- `worklogs = ARRAY( (author: Bob, timeSpent: 1h, startDate: Feb-1), (author: Alice, timeSpent: 2h, startDate: Feb-1), (author: Bob, timeSpent: 3h, startDate: Feb-2) )`

Then the following examples show grouping these work logs by author and by date:

- ```
  worklogs.GROUP($.author) →
  ARRAY(
    (group: Bob,   elements: ARRAY( (author: Bob, timeSpent: 1h, startDate: Feb-1), (author: Bob, timeSpent: 3h, startDate: Feb-2) )),
    (group: Alice, elements: ARRAY( (author: Alice, timeSpent: 2h, startDate: Feb-1) ))
  )
  ```
- ```
  worklogs.GROUP($.startDate) →
  ARRAY(
    (group: Feb-1, elements: ARRAY( (author: Bob, timeSpent: 1h, startDate: Feb-1), (author: Alice, timeSpent: 2h, startDate: Feb-1) )),
    (group: Feb-2, elements: ARRAY( (author: Bob, timeSpent: 3h, startDate: Feb-2) ))
  )
  ```

The expressions above show the key-value map values using a pseudo-formula. It is used only to demonstrate the values; Expr currently does not support this language or any way to create arbitrary key-value maps.

INDEX_OF

INDEX_OF(A, Element)

Finds the first occurrence of an element in the array. The comparison is done in the same way function EQ (=) works.

Parameter	Type	Description
`A`	`Array`	Array to be searched.

Parameter	Type	Description
Element	Any	Element to search for.
→ Result	Integer	Returns an index of a first occurrence of a specified element in an array. If the element is not found, returns undefined.

Note: the array is zero-based, so the first element is at Index = 0.

Example:

- INDEX_OF(ARRAY(1,3,3,3,5), 3) → 1

INDEXES

INDEXES(A)

Creates an array of indexes of A, starting with 0. For example, an array with 3 elements would return ARRAY(0, 1, 2).

Parameter	Type	Description
A	Array	Array of elements.
→ Result	Array	Array of indexes, starting at 0.

Example:

- INDEXES(ARRAY("Cat","DOG","BIRD")) → ARRAY(0,1,2)

IS_EMPTY

IS_EMPTY(A)

Returns true if the array is empty.

Parameter	Type	Description
A	Array	Array of elements.
→ Result	Boolean	Returns true (1) if the array is empty; false (0) if the array is not empty. Note: will return true for undefined, but false for an empty text string ("").

Examples:

- IS_EMPTY(ARRAY("Cat","DOG","BIRD")) → 0
- IS_EMPTY(ARRAY()) → 1

JOIN

JOIN(A)

JOIN(A, Sep, Gs, Ge)

Produces a text string representing any value. If the given value is an array, the text representation is composed by converting each element to text (per **Text** parameter conversion) and joining them together using ", " as a separator. Then the joined text is put into group parentheses **Gs** and **Ge**. If an array element is an array itself, the procedure repeats recursively.

A non-default separator may be passed as the **Sep** parameter.

Parameter	Type	Description
`A`	`Any`	Value to convert to text.
`Sep` (Optional)	`Text`	Optional separator to replace ", ".
`Gs` (Optional)	`Text`	Optional separator to replace "(".
`Ge` (Optional)	`Text`	Optional separator to replace ")".
→ Result	`Text`	Text comprised of all values in the array.

Example:

- `JOIN(ARRAY("Cat","Dog","Bird"))` → `(Cat, Dog, Bird)`
- `JOIN(ARRAY(ARRAY("Cat","Dog","Bird")), ARRAY("Sheep", "Pig")), " + " , "{" , "}" )` → `{{Cat + Dog + Bird} + {Sheep + Pig}}`
- `JOIN("Cat")` → `(Cat)`

LAST

LAST(A)

Returns the last element of the array, or undefined if the array is empty.

Parameter	Type	Description
`A`	`Array`	Array of any type.
→ Result	`Any`	Last element contained in the array. If empty, returns undefined.

Example:

- `LAST(ARRAY(1, 2, 3))` → `3`

LAST_INDEX_OF

LAST_INDEX_OF(A, Element)

Finds the last occurrence of an element in the array.

Parameter	Type	Description
`A`	`Array`	Array to be searched.
`Element`	`Any`	Element to search for.
→ Result	`Integer`	Returns an index of a last occurrence of a specified element in an array. If the element is not found, returns undefined.

Note: the array is zero-based, so the first element is at Index = 0.

Example:

- `LAST_INDEX_OF(ARRAY(1,2,2,2,3), 2) → 3`

MAP

MAP(A, $)

Applies the user function to every element of the array.

Parameter	Type	Description
`A`	`Array`	Array of elements to be mapped.
`$`	`User Function`	Function to be applied to each element.
→ Result	`Array`	Array containing the results.

Example:

- `ARRAY(1, 2, 3).MAP(x -> x * 100) → ARRAY(100, 200, 300)`
- `affectedVersions.MAP($.releaseDate - $.startDate)`

MERGE_ARRAYS

MERGE_ARRAYS(Array1, Array2, ..., ArrayN)

Produces a single array with the elements of all parameter arrays. Equal to `ARRAY(a1, a2, ...).FLATTEN()`.

Parameter	Type	Description
`Array1, Array2, ..., ArrayN`	`Array`	Arrays to be grouped.

Parameter	Type	Description
→ Result	`Array`	Single array containing all elements.

Example:

- `MERGE_ARRAYS(ARRAY(1, 2, 3),ARRAY(4,5,6),ARRAY(7)) → ARRAY(1,2,3,4,5,6,7)`

NONE

NONE(A, $)

Checks that **$** returns false for all elements in the array.

Parameter	Type	Description
`A`	`Array`	Array of elements to be used in $.
`$`	`User Function`	Function to be applied to each element.
→ Result	`Boolean`	Returns `true` (1) or `false` (0). If the array is empty, returns `true` (1).

This function is the inverse of ANY()

Examples:

- `NONE(fixVersions, $`[56]`.startDate < TODAY())`→ Returns true if no fixVersions started before the current date.

RECURSIVE_FLATTEN

RECURSIVE_FLATTEN(A)

Performs recursive flattening *and compacting* of the array. The resulting array is guaranteed to be flat and not contain undefined values.

Parameter	Type	Description
`A`	`Array`	Array of arrays.
→ Result	`Array`	A single array, containing all elements with no undefined values.

Examples:

- `RECURSIVE_FLATTEN(ARRAY(ARRAY(1, undefined, 2), ARRAY(2, 3), 100)) → ARRAY(1, 2, 2, 3, 100)`

56 http://expr.pocker.almworks.com/secure/StructureBoard.jspa?s=3

REDUCE

REDUCE(A, $)

Reduces an array to a single value based on **$**.

Parameter	Type	Description
A	Array	Array to be reduced.
$	User Function	Function containing two parameters.
→ Result	Any	• Applies the function to the first two elements in the array. Then applies the function again, using the resulting value and the third element. And so on. • For an empty array, returns undefined. • For an array with one element, returns that element.

Examples:

- `ARRAY(2, 3, 2, 1, 2).REDUCE((a, b) -> a * b) → 24`

REVERSE

REVERSE(A)

Reverses the order of elements in the array.

Parameter	Type	Description
A	Array	Array of elements.
→ Result	Array	Array with elements in reverse order.

Example:

- `REVERSE(ARRAY(1, 2, 3, 4)) → ARRAY(4, 3, 2, 1)`

SEQUENCE

SEQUENCE(from, to)

Creates an array of integer numbers, starting with **from** and ending with **to** (inclusive). If **to** is less than **from**, the sequence will be descending.

Parameter	Type	Description
from	Integer	Starting integer.

Parameter	Type	Description
to	Integer	Ending integer.
→ Result	Array	Array of integers.

Examples:

- SEQUENCE(3, 6) → ARRAY(3, 4, 5, 6)
- SEQUENCE(6, 3) → ARRAY(6, 5, 4, 3)

SIZE

SIZE(A)

Returns the number of elements in the array.

Parameter	Type	Description
A	Array	Array of elements.
→ Result	Integer	Number of elements contained in the array.

Example:

- SIZE(ARRAY(1, 2, 3, 4)) → 4
- SIZE(ARRAY(1, ARRAY(2, 3, 4), undefined)) → 3

SORT

SORT(A)

Sorts the array using a natural order of elements: numbers, text values, item values, and then array values (which are compared with respect to all the non-array elements). Item values are compared first by item type (lexicographically, so a user would come before a version because "u" comes before "v") and then either by the item's natural order, if it exists, or by the item's text representation, if it doesn't.

Parameter	Type	Description
A	Array	Array of elements to be sorted.
→ Result	Array	Sorted array.

Example:

- SORT(ARRAY(3,1,2)) → ARRAY(1,2,3)

SORT_BY

SORT_BY(A, $)

Sorts the array by comparing the values produced by calling the user function **$** for each array element. Results of the user function **$** calls are compared the same way as the 2022-12-15_19-52-26_.Expr Function Reference vFUTURE-dc#SORT(see page 346) function.

Parameter	Type	Description
`A`	`Array`	Array of elements to be sorted.
`$`	`User Function`	Function to be applied to each element.
→ Result	`Array`	Sorted array.

Example:

- `SORT_BY(fixVersions, $.releaseDate)`

SUBARRAY

SUBARRAY(A, from, to)

Produces an array with elements from the given array. Parameters from (inclusive) and to (exclusive) define the range. Indexes are zero-based.

Parameter	Type	Description
`A`	Array	Array of elements.
`from`	`Integer`	Starting index (inclusive).
`to`	`Integer`	Ending index (exclusive).
→ Result	`Array`	Array containing elements between `from` and `to`.

Examples:

- `SUBARRAY(ARRAY("Cat", "Dog", "Mouse", "Bird", "Sheep"), 1, 3)` → `ARRAY("Dog", "Mouse")`

UNIQUE

UNIQUE(A)

Removes duplicates from the array. The order of non-duplicate elements is preserved.

Parameter	Type	Description
`A`	`Array`	Array of elements.

Parameter	Type	Description
→ Result	`Array`	Array with duplicates removed.

Note: the equality is strict, so "0" and 0 would be different elements.

Example:

- `UNIQUE(ARRAY(1, 2, 1, 3, 3, 4))` → `ARRAY(1, 2, 3, 4)`

WITHOUT

WITHOUT(A, Value)

Returns a new array with all the elements from the input array except those equal to **Value**.

Parameter	Type	Description
`A`	`Array`	Array of elements.
`Value`	`Any`	Element to be removed.
→ Result	`Array`	New array with `Value` removed.

Equality is done using the same logic as the = operator and CONTAINS.

Example:

- `WITHOUT(ARRAY(1, 2, 1, 3, 3, 4), 1)` → `ARRAY(2, 3, 3, 4)`

Conditional Functions

CASE

`CASE(Value, Match1, Result1, Match2, Result2, ..., Default)`

Checks if the `Value` matches against several checks and returns a corresponding result.

Parameter	Type	Description
`Value`	`Text/ Joined`	Value to check.
`Match1, Match2, ..., MatchN`	`Text/ Joined`	Text patterns to check against. The first matching pattern will define the result. A pattern can be an exact value, a wildcard expression or a regular expression. See Expr Pattern Matching(see page 417) for details.
`Result1, Result2, ..., ResultN`	`Any`	Values to return from the function, each value corresponds to the preceding `Match` parameter.

Parameter	Type	Description
`Default` (Optional)	`Any`	Optional default value, to be returned if none of the patterns match. If not specified, `undefined` is returned.
→ Result	`Any`	Result1, Result2, etc. depending on which pattern matched, or Default, or *undefined*.

This function is typically used to map text values to numbers.

Examples:

- `CASE(Priority; "Highest"; 10; "High"; 5; "Medium"; 3; 1)`
- `CASE(Version; "V1*"; 1; "V2*"; 2)`

If the `Value` is `undefined`, the function immediately returns the `Default` result (or `undefined` if there's no default), so there is usually no need to use `undefined` as one of the matches.

CHOOSE

`CHOOSE(Index; Value1; Value2; ...)`

Based on the value of `Index`, returns the corresponding value from the argument list.

Parameter	Type	Description
`Index`	`Number`	Numeric index, with 1 corresponding to `Value1`, 2 corresponding to `Value2` and so on.
`Value1`, `Value2`, ..., `MatchN`	`Any`	The values to pick from.
→ Result	`Any`	The Value corresponding to Index.

Examples:

- `CHOOSE(1; "A"; "B"; "C")` → `"A"`
- `CHOOSE(2; "A"; "B"; "C")` → `"B"`

DEFINED

`DEFINED(Value)`

Checks if the value is defined.

Parameter	Type	Description
`Value`	`Any`	Value to check.
→ Result	`Boolean`	Returns false (0) if `Value` is `undefined` and true (1) otherwise.

Example:

- `IF(DEFINED(Resolution); ...)`

DEFAULT

`DEFAULT(Value; DefaultValue)`

Substitutes `DefaultValue` if the `Value` is `undefined`.

Parameter	Type	Description
`Value`	`Any`	Value to check.
`DefaultValue`	Any	Value to be returned if `Value` is `undefined`.
→ Result	`Any`	If `Value` is defined, returns `Value`. Otherwise, returns `DefaultValue`.

Examples:

- `DEFAULT(100; 500)` → `100`
- `DEFAULT(undefined; 500)` → `500`

IF

`IF(Condition1; Result1; Condition2; Result2; ...; Default)`

Checks one or several conditions and returns the result associated with the first true condition.

Parameter	Type	Description
`Condition1`, `Condition2`, ..., `Condition3`	`Any`	Value to check. The values are evaluated using "truthfulness check" – the first value that is "truthy" (not undefined, not zero and not an empty string), will define the returned value.
`Result1`, `Result2`, ..., `ResultN`	`Any`	Results to be returned, each result corresponding to the preceding check.
`Default` (Optional)	`Any`	Optional default value, to be returned if none of the patterns match. If not specified, `undefined` is returned.
→ Result	`Any`	Result1, Result2, etc. depending on which pattern matched, or Default, or *undefined*.

Examples:

- `IF(Estimate > 0; Duration / Estimate; 0)`
- `IF(N = 0; "No apples"; N = 1; "One apple"; CONCAT(N; " apples"))`

IFERR

`IFERR(Value; FallbackValue)`

Checks if calculating `Value` produced an error and substitutes `FallbackValue` instead of the error value.

Parameter	Type	Description
`Value`	`Any`	Value to check.
`FallbackValue`	`Any`	Value to be returned if calculating `Value` produces an error.
→ Result	`Any`	If `Value` calculated without an error, returns `Value`. Otherwise, returns `DefaultValue`.

Normally, if an error occurs while calculating a formula, it is propagated upwards, and the result of the whole expression will be an error. This function helps circumvent that.

Example:

- `IFERR(100 / 0; 100) → 100`

ISERR

`ISERR(Value; ErrorCode)`

Checks if calculating value produced an error.

Parameter	Type	Description
`Value`	`Any`	Value to check.
`ErrorCode` (Optional)	`Integer`	Optional error code. See Expr Error Codes(see page 414) for a list.
→ Result	`Boolean`	Returns true (1) if there was an error. If `ErrorCode` is specified, returns true only if the error was of the specified error code.

Examples:

- `ISERR("Ham") → 0`
- `ISERR(1 / 0) → 1`
- `ISERR(1 / 0, 4) → 1  //Note: Error code 4 is an Arithmetic Error`

Duration Functions

Duration is represented as a number of milliseconds. To create a value or make sense of a value, you need one of the following functions to convert a string to a duration and vice versa.

You can add duration to a date or date/time value and treat the result as a new date/time, but only if it's a calendar duration. This does not work with work duration.

To understand why, let's consider you wanted to add 16 hours at a date or date/time. The result should be slightly less than a day later. However, when using work duration, adding 16 hours will result in a date at least 2 days later (maybe more, if it crosses a weekend), based on Jira's default 8h/day 5-day work week.

CALENDAR_DAYS

`CALENDAR_DAYS(Duration)`

Returns a number of calendar days represented by the duration value as a decimal number.

Parameter	Type	Description
`Duration`	`Text/Each`	Value expressed in Jira Duration format (1d 3h 30m).
→ Result	`Number`	Number of calendar days. May return a fractional number of days.

Examples:

- `CALENDAR_DAYS(DURATION("10d")) → 10`
- `CALENDAR_DAYS(DURATION("12h")) → 0.5`

CALENDAR_HOURS

`CALENDAR_HOURS(Duration)`

Returns a number of hours represented by the duration value as a decimal number.

Parameter	Type	Description
`Duration`	`Text/Each`	Value expressed in Jira Duration format (1d 3h 30m).
→ Result	`Number`	Number of hours. May return a fractional number of hours.

Examples:

- `CALENDAR_HOURS(DURATION("10d")) → 240`
- `CALENDAR_HOURS(DURATION("12h 45m")) → 12.75`

CALENDAR_MINUTES

`CALENDAR_MINUTES(Duration)`

Returns a number of minutes represented by the duration value as a decimal number.

Parameter	Type	Description
`Duration`	`Text/Each`	Value expressed in Jira Duration format (1d 3h 30m).
→ Result	`Number`	Number of minutes. May return a fractional number of minutes.

Example:

- `CALENDAR_MINUTES(DURATION("3h")) → 180`

CALENDAR_SECONDS

`CALENDAR_SECONDS(Duration)`

Returns a number of seconds represented by the duration value as a decimal number.

Parameter	Type	Description
`Duration`	`Text/Each`	Value expressed in Jira Duration format (1d 3h 30m).
→ Result	`Number`	Number of calendar seconds. May return a fractional number of seconds.

Example:

- `CALENDAR_SECONDS(DURATION("1h")) → 3600`

CALENDAR_DURATION

`CALENDAR_DURATION(Start, Finish, Calendar)`

Returns the time spent between start and finish according to the chosen calendar. Be aware that calendars supplied by Structure do not depend on the user's time zone (system time zone is used), but some functions (like DATETIME) do. Other calendars may or may not depend on the user's time zone depending on the implementation.

Parameter	Type	Description
`Start`	`Date`	Date or date/time value to start counting calendar time.
`Finish`	`Date`	Date or date/time value to finish counting calendar time.
`Calendar (Optional)`	`Text`	Calendar name in current user locale. If omitted, the default (24/7) calendar is used, and the result would be equal to (Finish - Start).
→ Result	`Number`	Duration in milliseconds.

Example:

- `CALENDAR_DURATION(DATE("18/Jul/2022"), DATE("19/Jul/2022"), "Standard work calendar 8/5") → 28800000 (8h in milliseconds)`

DURATION

`DURATION(Text)`

Converts a text representation of a calendar duration to a number.

This function ignores Jira's settings for work time, so `DURATION("1w") = DURATION("7d")` and `DURATION("1d") = DURATION("24h")`.To consider work time, use JIRA_DURATION(see page 365).

Parameter	Type	Description
Duration	Text/Each	Value expressed in Jira Duration format (1d 3h 30m).
→ Result	Number	Duration in milliseconds.

Examples:

- `DURATION("1w 2d 3h 4m")`
- `DURATION("3d")`

FORMAT_DURATION

`FORMAT_DURATION(Duration)`

Converts duration value to the Jira format with numbers followed by symbols specifying the time unit.

Parameter	Type	Description
Duration	Number/Each	Duration value in number format.
→ Result	Number	Duration value converted to Jira Duration format (1d 3h 30m).

Example:

- `FORMAT_DURATION(DURATION("1w 1d")) → "1w 1d"`

JIRA_DAYS

`JIRA_DAYS(Duration)`

Returns a number of work days in the specified duration according to Jira's settings.

Parameter	Type	Description
Duration	Text/Each	Value expressed in Jira Duration format (1d 3h 30m).
→ Result	Number	Number of work days. (By default, one day is 8 hours.) May return a fractional number.

Example:

- `JIRA_DAYS(DURATION("24h")) → 3`
- `JIRA_DAYS(DURATION("12h")) → 1.5`

JIRA_DURATION

`JIRA_DURATION(Text)`

Converts a text representation of a Jira work duration to a number.

The specified time is work time, according to Jira's settings. With the default Jira settings, `JIRA_DURATION("1w") = JIRA_DURATION("5d")` and `JIRA_DURATION("1d") = JIRA_DURATION("8h")`. To use calendar time, use DURATION(see page 364).

Parameter	Type	Description
`Duration`	`Text/Each`	Value expressed in Jira Duration format (1d 3h 30m).
→ Result	`Number`	Duration in milliseconds.

Examples:

- `JIRA_DURATION("1w 2d 3h 4m")`
- `JIRA_DURATION("3d")`

JIRA_WEEKS

`JIRA_WEEKS(Duration)`

Returns a number of work weeks in the specified duration according to Jira's settings.

Parameter	Type	Description
`Duration`	`Text/Each`	Value expressed in Jira Duration format (1d 3h 30m).
→ Result	`Number`	Number of work days. (By default, one week is 5 work days.) May return a fractional number.

Example:

- `JIRA_WEEKS(JIRA_DURATION("10d")) → 2`
- `JIRA_WEEKS(DURATION("5d")) → 3`

Numeric Functions

ABS

`ABS(Value)`

Calculates the absolute value of a number.

Parameter	Type	Description
`Value`	`Number/Each`	Value to check.
→ Result	`Number`	Absolute value of `Value`.

Examples:

- `ABS(5) → 5`
- `ABS(-4) → 4`

CEILING

`CEILING(Value; N)`

Rounds value up to the N^{th} decimal place.

Parameter	Type	Description
`Value`	`Number/Each`	Number to round.
`N` (Optional)	`Integer`	How many decimal places to round up to. Negative numbers round up to tens, hundreds, etc. Default value: `0` (round to an integer).
→ Result	`Number`	`Value` rounded up to the `N`th place.

Examples:

- `CEILING(1.678)` → `2`
- `CEILING(12.34; 1)` → `12.4`
- `CEILING(12.34; -1)` → `20`
- `CEILING(-3.14)` → `-3`

FLOOR

`FLOOR(Value; N)`

Rounds value down to the `N`th decimal place.

Parameter	Type	Description
`Value`	`Number/Each`	Number to round.
`N` (Optional)	`Integer`	How many decimal places to round down to. Negative numbers round down to tens, hundreds, etc. Default value: `0` (round to an integer).
→ Result	`Number`	`Value` rounded down to the `N`th place.

Examples:

- `FLOOR(1.678)` → `1`
- `FLOOR(12.34; 1)` → `12.3`
- `FLOOR(17.34; -1)` → `10`
- `FLOOR(-3.14)` → `-4`

LN

`LN(x)`

Returns the logarithm for 'x' with base of e.

Parameter	Type	Description
x	`Number`	Value to check
→ Result	`Number`	logarithm for x with a base of e

LOG

`LOG(x), LOG(x,b)`

LOG(x) returns the logarithm for 'x' with base of 10. LOG(x,b) returns the logarithm for 'x' with base of 'b'.

Parameter	Type	Description
x	`Number`	Value to check
b (Optional)	`Number`	Base value. If omitted or an empty value is entered, uses base 10.
→ Result	`Number`	logarithm for x

Example - the following will categorize projects based on the logarithm of all the story points within them:

```
WITH projectSize = LOG(SUM{storypoints}):
IF(
projectSize < 1; "Small";
projectSize < 2; "Medium";
projectSize >= 2; "Large"
)
```

LOG10

`LOG10(x)`

Returns the logarithm for 'x' with base of 10. Same as LOG with only an x variable.

Parameter	Type	Description
x	`Number`	Value to check
b (Optional)	`Number`	Base value
→ Result	`Number`	logarithm for x

MOD

`MOD(A; N)`

Returns the remainder from dividing A by N.

Parameter	Type	Description
A	Integer/Each	The dividend, must be an integer.
N	Integer	The divisor, must be an integer.
→ Result	Number	The remainder from dividing A by N.

Example:

- MOD(17; 5) → 2

MUL

MUL(Value1, Value2,...)

MUL(A)

Short for "multiply" - produces the product of all values passed as arguments. When used with an array, produces the product of all values in the array.

Parameter	Type	Description
Value1, Value2, ..., ValueN OR A	Number Array	Series of number values. Array containing numeric elements.
→ Result	Integer	Product of all numeric elements.

Undefined values are ignored. Non-numeric values result in an error.

Example:

- MUL(2, 3, 5) → 30
- MUL(ARRAY(1, 2, 3, 4)) → 24

NUMBER

NUMBER(Value, DefaultOpt)

Converts value to number. This function is rarely needed, because conversion to number happens automatically when needed.

Parameter	Type	Description
Value	Any	Value to convert
Default (Optional)	Number	Optional. If provided and Value cannot be converted to a number, this function returns the Default rather than an error.

Parameter	Type	Description
→ Result	`Number`	`Value` converted to a number.

Example:

- `NUMBER("1.234") → 1.234`

POW

`POW(B; E)`

Produces B to the power of E (B^E). Both values can be fractional.

Parameter	Type	Description
`B`	`Number/Each`	Base
`E`	`Number`	Exponent
→ Result	`Number`	B to the power of E (B^E)

Example:

- `POW(3; 3) → 27`
- `POW(27; 1/3) → 3`

ROUND

`ROUND(Value, N)`

Rounds value to the N^{th} decimal place.

Parameter	Type	Description
`Value`	`Number/Each`	A number to round.
`N` (Optional)	`Integer`	How many decimal places to round to. Negative numbers round to the nearest tens, hundreds, etc. Default value: `0` (round to an integer).
→ **Result**	`Number`	`Value` rounded to the N^{th} place.

Examples:

- `ROUND(1.678) → 2`
- `ROUND(12.34, 1) → 12.3`
- `ROUND(12.34, -1) → 10`
- `ROUND(ARRAY(1.1, 2.6)) → ARRAY(1, 3)`

SIGN

`SIGN(Value)`

Returns the sign of the Value (`1` for positive, `-1` for negative).

Parameter	Type	Description
`Value`	`Number/Each`	Value to check.
→ **Result**	`Number`	Returns `1` if `Value` is positive, `-1` if `Value` is negative.

Examples:

- `SIGN(123)` → `1`
- `SIGN(0)` → `0`
- `SIGN(-123)` → `-1`

SQR

`SQR(Value)`

Returns the passed numerical value, squared.

Parameter	Type	Description
`Value`	`Number/Each`	Numerical value.
→ **Result**	`Number`	$Value^2$

Example:

- `SQR(5)` → `25`

SQRT

`SQRT(Value)`

Returns the square root of the passed numerical value.

Parameter	Type	Description
`Value`	`Number/Each`	Numerical value.
→ **Result**	`Number`	$\sqrt{Value}$

Example:

- `SQRT(25)` → `5`

SUM

`SUM(Number1, Number2, ...)`

`SUM(A)`

Produces the total of all numeric values passed as arguments. When used with an array, produces a total of all numeric elements in the array.

Parameter	Type	Description

`Number1, Number2, ..., NumberN)`
 	
OR
 	
`A`
	`Number`
	
	`Array`
		Array containing numeric elements.

→ Result	`Integer`	Sum of all numeric elements.

Undefined values are ignored. Non-numeric values (that cannot be converted to numbers) result in an error.

Example:

- `SUM(1; 3; 5)` → `9`
- `SUM(ARRAY(1, 2, 3, 4))` → `10`

Statistical Functions

AVERAGE

AVERAGE(Number1, Number2, ...)

AVERAGE(A)

Calculates the average of numbers in an array, or a series of numbers. When used with multiple arguments, finds the average of the numbers passed as parameters. When used with an array, finds the average of the numbers in the array.

Parameter	Type	Description

`Number1, Number2, ..., NumberN`
 	
OR
 	
`A`
	`Number`
	
	`Array`
		Series of number values.
		
		Array of numbers values to be considered.

→ Result	`Number`	Average of the numbers. If the array is empty, or all element are undefined, returns undefined.

Undefined values are ignored. Values that cannot be converted to numbers produce an error.

Examples:

- `AVERAGE(1; 3; 5)` → `3`
- `AVERAGE(numberArray)`

MAX

MAX(Number1, Number2, ...)

MAX(A)

When used with multiple arguments, finds the largest value among the numbers passed as parameters. When used with an array, finds the maximum number in the array.

Parameter	Type	Description

`Number1, Number2, ..., NumberN`
 	
OR
 	
`A`
	`Number`
	
	`Array`
		Series of number values.
		
		Array of numbers values to be considered.

→ Result	`Number`	Largest value. If the array is empty, or all element are undefined, returns undefined.

Undefined values are ignored. Values that cannot be converted to numbers produce an error.

Examples:

- `MAX(due_date; updated_date)`
- `MAX(0; -10; undefined; 10)` → `10`
- `MAX(ARRAY(1,6,3))` → `6`

MEDIAN

MEDIAN(A)

Calculates the median of the numbers in an array. Equal to **PERCENTILE(A, 0.5)**.

Parameter	Type	Description
`A`	`Array`	Array of numbers values to be considered.
→ Result	`Number`	Median value.

Example:

- `MEDIAN(ARRAY(1,2,5,7,8))` → `5`

MIN

MIN(Number1, Number2, ...)

MIN(A)

When used with multiple arguments, find the smallest value among the numbers passed as parameters. When used with an array, finds the minimum number in the array.

Parameter	Type	Description

`Number1, Number2, ..., NumberN`
 	
OR
 	
`A`
	`Number`
	
	`Array`
		Series of number values.
		
		Array of numbers values to be considered.

→ Result	`Number`	Smallest value. If the array is empty, or all element are undefined, returns undefined.

Undefined values are ignored. Values that cannot be converted to numbers produce an error.

Examples:

- `MIN(0; -10; undefined; 10)` → `-10`
- `MAX(ARRAY(1,6,3))` → `1`

PERCENTILE

PERCENTILE(A, N)

Calculates `N` percentile of the values in the given array.

Parameter	Type	Description
`A`	`Array`	Array of values to be considered.
`N`	`Number`	Value between 0.0 and 1.0. Any other value will produce an error.
→ Result	Number	Resulting value.

Undefined elements are ignored. Non-numeric values that cannot be converted to a number will result in an error.

If `A` contains only numbers, PERCENTILE(A, 0) is equal to A.MIN() and PERCENTILE(A, 1) is equal to A.MAX().

Example:

- `PERCENTILE(ARRAY(1,2,3,4,5), 0.25)` → `2`

QUARTILE

QUARTILE(A, N)

Calculates the quartile of the numbers in an array, or a series of numbers. Equal to **PERCENTILE(A, N*0.25)**.

Parameter	Type	Description
`A`	`Array`	Array of values to be considered.
`N`	`INT`	Interger value between 0 and 4. 0 = value at the 0 percentile 1 = value at the 25th percentile 2 = value at the 50th percentile 3 = value at the 75th percentile 4 = value at the 100th percentile
→ Result	Number	Quartile value.

Example:

- `QUARTILE(ARRAY(1,2,3,4,5), 3) → 4`

STDEV

STDEV(A)

Calculates standard deviation, based on a sample population. *For the entire population, use STDEVP.*

Parameter	Type	Description
`A`	`Array`	Array of numbers values to be considered.
→ Result	`Number`	Standard deviation.

Example:

```
STDEV(ARRAY(1,2,3))  ->  1
```

STDEVP

STDEVP(A)

Calculates standard deviation, based on the entire population. *For a sample of the population, use STDEV.*

Parameter	Type	Description
`A`	`Array`	Array of numbers values to be considered.

Parameter	Type	Description
→ Result	`Number`	Standard deviation.

Example:

```
STDEVP(ARRAY(1,2,3))  -> 0.8165
```

UMAX

UMAX(Value1, Value2, ...)

UMAX(A)

Same as MAX(see page 373), but does a universal comparison, accepting any type of values, including items and arrays. When used with multiple arguments, finds the largest value passed as parameters. When used with an array, finds the maximum value in the array.

Parameter	Type	Description

`Value1, Value2, ..., ValueN`
 	
OR
 	
`A`
	`Any`
	
	`Array`
		Series of values.
		
		Array of values to be considered.

→ Result	`Any`	Largest value. If the array is empty, or all element are undefined, returns undefined.

Undefined values are ignored.

Examples:

- `UMAX("aardvark", "zebra", "lion")` → `"zebra"`
- `UMAX(ArrayofAnyTypes)`

UMAX_BY

UMAX_BY(A, $)

Returns the maximum value by comparing all values in the array using the values calculated by calling **$** for each element. Applies universal comparison to the value (UMAX(see page 376)).

Parameter	Type	Description
`A`	`Array`	Array of values to be considered.
`$`	`User Function`	

Parameter	Type	Description
→ Result	`Any`	Maximum value. If the array is empty, or all element are undefined, returns undefined.

Undefined values are ignored.

Example:

`fixVersions.UMAX_BY($.releaseDate)` → returns the latest fix version

UMIN

UMIN(Number1, Number2, ...)

UMIN(A)

Same as MIN, but does a universal comparison, accepting any type of values, including entities. When used with multiple arguments, find the smallest value passed as parameters. When used with an array, finds the minimum value in the array.

Parameter	Type	Description

`Number1, Number2, ..., NumberN`
 	
OR
 	
`A`
	`Any`
	
	`Array`
		Series of values.
		
		Array of values to be considered.

→ Result	`Any`	Smallest value. If the array is empty, or all element are undefined, returns undefined.

Undefined values are ignored.

Examples:

- `UMIN("aardvark", "zebra", "lion")` → `"aardvark"`
- `UMIN(ArrayofAnyTypes)`

UMIN_BY

UMIN_BY(A, $)

Returns the minimum value by comparing all values in the array using the values calculated by calling **$** for each element. Applies universal comparison to the value (U(see page 372)MIN(see page 377)).

Parameter	Type	Description
`A`	`Array`	Array of values to be considered.

Parameter	Type	Description
$	User Function	
→ Result	Any	Minimum value. If the array is empty, or all element are undefined, returns undefined.

Undefined values are ignored.

Example:

`fixVersions.UMIN_BY($.releaseDate)` → returns the earliest fix version

Text Functions

Text functions let you manipulate character strings.

If a function expects a string but encounters a number, it converts it to a string using mathematical notation ("." decimal separator, no thousands separator).

CONCAT

`CONCAT(Value; ...)`

Concatenates (merges) text values into a single text.

Parameter	Type	Description
Value1, Value2, ..., ValueN	Text/Joined	Text string(s). Accepts any number of arguments. Ignores falsy(see page 305) values.
→ Result	Text	A single text containing all the values combined.

Example:

- `CONCAT(Reporter; ' => '; Assignee)`

EXACT

`EXACT(A; B)`

Checks if text value A is exactly the same as text value B.

Parameter	Type	Description
A	Text/Joined	Text value.
B	Text/Joined	Text value.
→ Result	Boolean	Returns true (1) if values are exactly the same. Otherwise, false (0).

This comparison is case sensitive, which is different from comparing A with B using an equals sign or text matching. Undefined values will be equal to each other and to empty strings.

Examples:

- `EXACT("Fox"; "fox") → 0`
- `EXACT("Fox"; "Fox") → 1`
- `EXACT(""; undefined) → 1`

LEFT

`LEFT(Value; N)`

Returns up to N leftmost characters from a text value.

Parameter	Type	Description
`Value`	`Text/Joined`	Text to get characters from.
`N`	`Integer`	The number of characters to get.
→ Result	`Text`	The first `N` characters, starting from the left. If `Value` contains fewer characters, all of them are returned. If the value is less than zero, an empty text is returned.

Example:

- `LEFT("abc"; 2) → "ab"`

LEN

`LEN(Value)`

Returns the number of characters in a text value.

Parameter	Type	Description
`Value`	`Text/Joined`	Text to count. If the value is not text, it is converted to text first.
→ Result	`Integer`	The number of characters in `Value`.

Example:

- `LEN("abc") → 3`

LOWER

`LOWER(Value)`

Converts text to lowercase. The locale of the current user is applied.

Parameter	Type	Description
`Value`	`Text/Each`	Text to convert.

Parameter	Type	Description
→ Result	`Text`	`Value` in all lowercase.

Example:

- `LOWER("HAM") → "ham"`

MATCH

`MATCH(Value; Pattern)`

Checks if the Value matches the Pattern.

Parameter	Type	Description
`Value`	`Text/Joined`	Value to check.
`Pattern`	`Text/Joined`	Pattern to check against. Can be an exact value, a wildcard expression or a regular expression. See Expr Pattern Matching(see page 417) for details.
→ Result	`Boolean`	Returns `true` (1) or `false` (0).

Examples:

- `MATCH("Apples"; "Oranges") → 0`
- `MATCH(" Blocker "; "blocker") → 1`
- `MATCH("Hamster"; "ham*") → 1`
- `MATCH("The Flight of the Bumblebee"; "/.light.*beer?/") → 1`

MID

`MID(Value; Index; Count)`

Retrieves a part of the text.

Parameter	Type	Description
`Value`	`Text/Joined`	The text value to get a substring from.
`Index`	`Integer`	The starting index of the part to retrieve, 1-based (first character is at index 1).
`Count`	`Integer`	The number of characters to retrieve.
→ Result	`Text`	Text containing `Count` number of characters, starting from `Index`.

Example:

- `MID("A quick brown fox"; 3; 5) → "quick"`

REPEAT

`REPEAT(Value; N)`

Produces a text that is a repetition of the string value N times.

Parameter	Type	Description
`Value`	`Text/Joined`	Text to repeat.
`N`	`Integer`	The number of repetitions.
→ Result	`Text`	The repeated text.

Examples:

- `REPEAT("ha"; 3)` → `"hahaha"`
- `REPEAT(123, 3)` → `"123123123"`

REPLACE

`REPLACE(Value; Pattern; Replacement)`

Replaces all occurrences of `Pattern` with `Replacement` and returns the new text.

Parameter	Type	Description
`Value`	`Text/ Joined`	The text to manipulate.
`Pattern`	`Text`	Pattern to find. Can be an exact value, a wildcard expression or a regular expression. See Expr Pattern Matching(see page 417) for details.
`Replacement` (Optional)	`Text`	An optional text to use instead of the matched parts. If omitted, the matched parts are removed.
→ Result	`Text`	`Value` with replacements.

Examples:

- `REPLACE("I like cats"; "CAT"; "DOG")` → `"I like DOGs"`
- `REPLACE("Can you read this?"; "/[aeuio]/")` → `"Cn y rd ths?"`

REPLACE_AT

`REPLACE_AT(Value; Index; Count; Replacement)`

Replaces a specific part of the Value with Replacement text and returns the value.

Parameter	Type	Description
`Value`	`Text/ Joined`	The text to manipulate.

Parameter	Type	Description
`Index`	`Integer`	The starting index of the part to replace, 1-based (first character is 1, second is 2, etc.)
`Count`	`Integer`	The number of characters to replace. When Count is 0, the Replacement string gets inserted at the Index position.
`Replacement` (Optional)	`Text`	An optional text to use instead of the replaced part. If omitted, the part will be deleted.
→ Result	`Text`	`Value` with replacements.

When the values of Index and Count are out of range, they are brought to the nearest sensible value.

Examples:

- `REPLACE_AT("A"; 1; 1; "B")` → `"B"`
- `REPLACE_AT("What does the fox say?"; 6; 4; "did")` → `"What did the fox say?"`
- `REPLACE_AT("A step for mankind"; 3; 0; "small ")` → `"A small step for mankind"`
- `REPLACE_AT("A step for mankind"; 7; 1000)` → `"A step"`

RIGHT

`RIGHT(Value; N)`

Returns up to N rightmost characters from a string value.

Parameter	Type	Description
`Value`	`Text/ Joined`	Text to get characters from.
`N`	`Integer`	The number of characters to get.
→ Result	`Text`	The first `N` characters, starting from the right. If `Value` contains fewer characters, all of them are returned.

Example:

- `RIGHT("abc"; 2)` → `"bc"`

SEARCH

`SEARCH(Pattern; Value; Index)`

Finds the first occurrence of a pattern in the value.

Parameter	Type	Description
`Pattern`	`Text`	The text or pattern to look for. Can be an exact value, a wildcard expression or a regular expression. See Expr Pattern Matching(see page 417) for details.

Parameter	Type	Description
`Value`	`Text/ Joined`	The text to search in.
`Index` (Optional)	`Integer`	Optional parameter that provides an index to start searching at.
→ Result	`Integer`	Returns the index of the matched part (1-based), or `undefined` if not found.

Examples:

- `SEARCH("ham"; "The Ham is for the Hamster"; 6)` → `20`
- `SEARCH("Jedi*"; "Return of the Jedi")` → `15`
- `SEARCH("/^Jedi/"; "Not the Jedi you're looking for")` → `undefined`

SPLIT

`SPLIT(Value; Separator)`

Produces an array from the value by splitting it using a separator.

Parameter	Type	Description
`Value`	`Text/ Joined`	The text to split.
`Separator`	`Text`	The text or pattern to split by. Can be an exact value, a wildcard expression or a regular expression. See Expr Pattern Matching(see page 417) for details.
→ Result	`Array`	Returns an array which contains the split texts.

Examples:

- `SPLIT("One, Two, Three", ",")` → `ARRAY("One", "Two", "Three")`
- `SPLIT("A and B or C", "/ and | or /")` → `ARRAY("A", "B", "C")`

SUBSTRING

`SUBSTRING(Value; From; To)`

Returns a substring, indicated by a starting index and ending index. Note that the indexes are 0-based, unlike in some other functions.

Parameter	Type	Description
`Value`	`Text/ Joined`	The text to take the part from.

Parameter	Type	Description
`From`	`Integer`	Starting index, inclusive, 0 means the first character, `LEN(Value)-1` means the last character.
`To` (Optional)	`Integer`	Optional ending index, exclusive - the character at this index will not be included. If omitted, the substring will include all characters up to the end of the `Value`.
→ Result	`Text`	Returns the portion of the text contained between `From` and `To`.

If `To` value is greater than the text length, all characters will be included. If `To` is less than `From`, an empty text is returned.

Examples:

- `SUBSTRING("Batman"; 0; 3)` → `"Bat"`
- `SUBSTRING("Batman"; 3)` → `"man"`

TEXT

`TEXT(Value)`

Converts value to text. This function is rarely needed, because conversion to text happens automatically when needed.

Parameter	Type	Description
`Value`	`Any (If text, Text/Joined)`	Value to convert
→ Result	`Text`	`Value` converted to text.

Example:

- `TEXT(1.234)` → `"1.234"`

TRIM

`TRIM(Value)`

Removes leading and trailing whitespace from the text.

Parameter	Type	Description
`Value`	`Text/Each`	The text to manipulate.
→ Result	`Text`	Returns `Value` without leading/trailing whitespace.

Example:

- `TRIM(" Batman ")` → `"Batman"`

UPPER

`UPPER(Value)`

Converts the string to uppercase. The locale of the current user is applied.

Parameter	Type	Description
`Value`	`Text/Each`	The text to manipulate.
→ Result	`Text`	Returns `Value` in all uppercase.

Example:

- `UPPER("ham")` → `"HAM"`

User-related Functions

USER_NAME

`USER_NAME(UserKey)`

Parameter	Type	Description
`UserKey`	`Item/Each`	UserKey value.
→ Result	`Text`	Username.

Returns the username specified by the user key.

USER_DISPLAY_NAME

`USER_DISPLAY_NAME(UserKey)`

Returns the user's full name, defined by the user key.

Parameter	Type	Description
`UserKey`	`Item/Each`	UserKey value.
→ Result	`Text`	User's full name.

USER_EMAIL

`USER_EMAIL(UserKey)`

Returns the user's email, defined by the user key.

Parameter	Type	Description
`UserKey`	`Item/Each`	UserKey value.
→ Result	`Text`	User's email address.

USER_IS_ACTIVE

`USER_IS_ACTIVE(UserKey)`

Returns 1 for active user, 0 for inactive user.

Parameter	Type	Description
`UserKey`	`Item/Each`	UserKey value.
→ Result	`Number`	1 for active user, 0 for inactive user.

Miscellaneous Functions

ACCESS

`ACCESS(Item, PropertyName)`

Tries to access a property/field of the item.

Parameter	Type	Description
`Item`	`Item`	Item containing the property or a key-value map.
`PropertyName`	`Text`	Property or field being accessed.
→ Result	`Any`	Value in `PropertyName`. Undefined if the `Item` is not an item or key-value map, or it does not have this property.

Example:

- `ACCESS(epic, "Story Points")`

HISTORICAL_VALUE

HISTORICAL_VALUE(Item, Field, Date)

Show value of the field at specified date

Parameter	Type	Description
`Item`	`Item`	Issue item; use "this" for current row.
`Field`	`Text`	Field being accessed
`Date`	`Date/Time`	Date or date/time value; use DATETIME function to convert into it: #DATETIME(see page 386)
→ `Result`	`Any`	Field value at specified date (current value will be used for future dates)

Example:

- HISTORICAL_VALUE(this, "cost", DATETIME("15/May/18 6:24 PM")) → 5.55
- HISTORICAL_VALUE(this, "cost", NOW()) → 5.55

ME

`ME()`

Returns the user item of the current user.

Example:

- `IF(ME() = "admin"; "You're admin!")`

URL_ENCODE

`URL_ENCODE(Value)`

Translates a text into application/x-www-form-urlencoded format.

Parameter	Type	Description
`Value`	`Text`	Value to convert
→ Result	`Text`	`Value` converted to application/x-www-form-urlencoded format.

Example:

- `URL_ENCODE("http://www.test.com`[57]`") → "http%3A%2F%2Fwww.test.com`[58]`"`

URL_DECODE

`URL_DECODE(Value)`

Decodes an application/x-www-form-urlencoded text.

Parameter	Type	Description
`Value`	`Text`	Value to convert
→ Result	`Text`	Decoded `Value`.

Example:

- `URL_DECODE("http%3A%2F%2Fwww.test.com`[59]`") → "http://www.test.com/"`

3.8.3.3 Aggregate Function Reference

All standard aggregate functions and available modifiers are listed on this page.

57 http://www.test.com/
58 http://2fwww.test.com/
59 http://2fwww.test.com/

An aggregate function call contains an expression in curly braces ("{}"), which is calculated for the item and all sub-items (or, in some cases, for another subset of related items in the structure), and then the resulting values are aggregated according to the meaning of the aggregate function.

Aggregation Functions

ARRAY

Produces an array of the defined values for the item and/or its sub-items.

Summary	X	ARRAY{x}
T1	1	1, 2, 2, 3
T1.1	2	2
T1.2	2	2, 3
T1.2.1	3	3

Accepts modifiers: #ancestors(see page 394), #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400), #fromLevel(see page 396), #toLevel(see page 401), #reverse(see page 399), #distinct(see page 395) (does not include duplicate values), #flatten(see page 396) (if function produces an array, includes the inner values, rather than the array), #compact(see page 395)

AVG

Avg calculates an average of the defined values for the item and/or its sub-items. The result for avg is generally the same as sum/count. It returns nothing if there are no defined values for {x}.

Summary	X	AVG{X}
T1	3	2
T1.1	2	2
T1.2		1
T1.2.1	1	1

If a certain issue (or another kind of item) is included multiple times in the sub-tree, the average value will include the value for that issue only *once*. This behavior can be overridden by using the #all modifier.

Accepts modifiers: #all(see page 393), #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400), #preceding(see page 398), #levels(see page 398) (together with #preceding), #baseLevel(see page 394) (together with #preceding).

COUNT

Count calculates a count of defined values (or truthy values, if the #truthy modifier is specified) for the item and/or its sub-items.

Summary	X	COUNT{X}
T1	3	3
T1.1	2	1
T1.2		1
T1.2.1	1	1

If a certain issue (or another kind of item) is included multiple times in the sub-tree, it will be counted only *once*. This behavior can be overridden by using the #all modifier.

Accepts modifiers: #all(see page 393), #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400), #truthy(see page 401), #preceding(see page 398), #levels(see page 398) (together with #preceding), #baseLevel(see page 394) (together with #preceding).

JQL

Returns true if the current row is an issue and it matches this JQL.

Summary	JQL{Issuetype=task}
T1	1
T2	1
T3	0
T4	0

This is not an aggregate function; it's an embedded query. We've included it here because it's easy to mistake for an aggregate function - both use curly braces { }.

To learn more, see Embedded Queries(see page 0).

JOIN

Join concatenates (joins) strings from the item and its parents (or other items, if modifiers are used).

- By default, it joins all parent string values from root to the self value.
- If the current row has children and the #subtree(see page 400) modifier is set, join appends the values for children, wrapping them into characters (braces by default).
- Wrapping characters can be set by #beforeChildren(see page 394) and #afterChildren(see page 393) (see example for #subtree(see page 400) to see how it works).

Summary	X	JOIN{X}
T1	3	3
T1.1	2	3, 2
T1.2		3, ?
T1.2.1	1	3, ?, 1

Accepts modifiers: #ancestors(see page 394), #subtree(see page 400), #children(see page 394), #leaves(see page 397), #strict(see page 400), #reverse(see page 399), #separator(see page 399), #beforeChildren(see page 394), #afterChildren(see page 393), #fromLevel(see page 396), #toLevel(see page 401), #distinct(see page 395).

MAX

Max returns the maximum defined value for the item and/or its sub-items. Numeric, date, duration and text fields can be compared. Text fields are compared lexicographically.

Summary	X	MAX{X}
T1	3	3
T1.1	2	2
T1.2		1
T1.2.1	1	1

If the formula produces an array, this will find the maximum element within that array.

Accepts modifiers: #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400).

MEDIAN

Produces the median value. Works the same as PERCENTILE(see page 391)#0.5.

Summary	X	MEDIAN{x}
T1	1	2.5
T1.1	2	2
T1.2	3	3.5
T1.2.1	4	4

Undefined are ignored. Non-number values result in an error.

Accepts modifiers: #ancestors(see page 394), #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400), #fromLevel(see page 396), #toLevel(see page 401)

MIN

Min returns the minimum defined value for the item and/or its sub-items. Numeric, date, duration and text fields can be compared. Text fields are compared lexicographically.

Summary	X	MIN{X}
T1	3	1
T1.1	2	2
T1.2		1
T1.2.1	1	1

If the formula produces an array, this will find the minimum element within that array.

Accepts modifiers: #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400).

PARENT

Parent extracts the value from the parent row or from an ancestor row by a specified level.

Summary	X	PARENT{X}
T1	3	
T1.1	2	3
T1.2		3
T1.2.1	1	

Using the #level modifier, you can specify which row to extract the value from:

- `PARENT#level=-1{}` extracts the value from the parent row (same as `PARENT{}`)

- `PARENT#level=-2{}` extracts the value from the grandparent row
- `PARENT#level=1{}` extracts the value from the root row

Accepts modifier: #level(see page 397).

PERCENTILE

Calculates a percentile value from the defined values for the item and/or its sub-items. The hierarchy of values is ignored – all subject values are treated equally.

Must include the modifier #p, followed by the desired percentile (p=0.95, p=95). p=1 interpreted as 1%.

Summary	X	PERCENTILE#p=95{x}
T1	1	3.85
T1.1	2	2
T1.2	3	3.95
T1.2.1	4	4

Undefined are ignored. Non-number values result in an error.

Accepts modifiers: #ancestors(see page 394), #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400), #fromLevel(see page 396), #toLevel(see page 401), #p (required)

QUARTILE1

Works the same as PERCENTILE(see page 391)#0.25.

Summary	X	QUARTILE1{x}
T1	1	1.25
T1.1	2	2
T2	1	1.5
T2.1	2	2.25
T2.1.1	3	3

Undefined are ignored. Non-number values result in an error.

Accepts modifiers: #ancestors(see page 394), #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400), #fromLevel(see page 396), #toLevel(see page 401)

QUARTILE3

Works the same as PERCENTILE(see page 391)#0.75.

Summary	X	QUARTILE3{x}
T1	1	1.75
T1.1	2	2
T2	1	2.5
T2.1	2	2.75
T2.1.1	3	3

Undefined are ignored. Non-number values result in an error.

Accepts modifiers: #ancestors(see page 394), #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400), #fromLevel(see page 396), #toLevel(see page 401)

SJQL

Returns true if the current row matches this S-JQL(see page 428).

Summary	SJQL{child of [Issuetype=Task]}
T1	0
T1.2.1	1
T3	0
T3.1	0

This is not an aggregate function; it's an embedded query. We've included it here because it's easy to mistake for an aggregate function - both use curly braces { }.

To learn more, see Embedded Queries(see page 0).

SUM

Sum calculates a numerical total for the values calculated for the item and/or its sub-items.

Summary	X	SUM{X}
T1	3	6
T1.1	2	2
T1.2		1
T1.2.1	1	1

Other variations of `SUM` allow different types of aggregation:

- `SUM{}` (the same as `SUM#subtree{}`) aggregates values from all descendants
- `SUM#children{}` aggregates values from direct children only
- `SUM#leaves{}` aggregates values from leaves
- `SUM#preceding{}` aggregates values from the preceding rows
- If the formula produces an array, calculates the total of the elements of that array

Note that when the value of the expression under aggregation is not numeric (and cannot be converted[60] to a number), it is ignored.

If a certain issue (or another kind of item) is included multiple times in the sub-tree, the sum will include the value for that issue only *once*. This behavior can be overridden by using the `#all` modifier.

Accepts modifiers: #all(see page 393), #children(see page 394), #leaves(see page 397), #preceding(see page 398), #strict(see page 400), #subtree(see page 400), #levels(see page 398) (together with #preceding), #baseLevel(see page 394) (together with #preceding).

VALUES

60 https://wiki.almworks.com/display/structure/Expr+Language#ExprLanguage-NumbersandTextStrings

Produces an array of all distinct values for the item and/or its sub-items.

Summary	X	VALUES{x}
T1	1	1, 2, 3
T1.1	2	2
T1.2	2	2, 3
T1.2.1	3	3

If a value is an array, considers each value in the array separately.

Accepts modifiers: #ancestors(see page 394), #children(see page 394), #leaves(see page 397), #strict(see page 400), #subtree(see page 400), #fromLevel(see page 396), #toLevel(see page 401)

Aggregation Modifiers

#afterChildren

Defines the exit separator between children and parent rows. This modifier has a string parameter. The default exit separator is:

- "(" - for #beforeChildren
- ")" - for #afterChildren

Example

Summary	X	Formula
T1	3	3<{2, ?<{1}>}>
T1.1	2	2
T1.2		?<{1}>
T1.2.1	1	1

```
JOIN#subtree#beforeChildren="<{"#afterChildren="}>"{X}
```

Can be used with: join(see page 389).

#all

Tells the aggregate function to include duplicate items. By defaults, functions that count values ignore duplicate items.

Example

Summary	X	SUM{X}	SUM#all{X}	COUNT{X}	COUNT#all{X}
T1		2	6	1	3
T1.1	2	2	2	1	1
T1.1	2	2	2	1	1
T1.1	2	2	2	1	1

```
SUM#all{X}
COUNT#all{X}
```

Can be used with: sum(see page 392), count(see page 388), avg(see page 388).

#ancestors

Only process ancestors of the current row. This is the default behavior for join(see page 389).

Can be used with: array(see page 388), join(see page 389), median(see page 390), percentile(see page 391), quartile1(see page 391), quartile3(see page 391), values(see page 392).

#baseLevel

Can be used with SUM(see page 392) with the `#preceding` modifier to specify at which level the accumulation will be reset and start over from zero. This allows for accruing independent cumulative values in different sub-trees.

- By default, baseLevel is 0, which means that the accumulation will never start over and will cover the whole structure.
- If baseLevel is not zero, then the accumulation will be restarted once it reaches a row at the base level or higher in the hierarchy. For example, if you have epics at the top level, and stories underneath them, `SUM#preceding#baseLevel=1{story_points}` will accumulate the Story Points within the scope of each epic independently.

Can be used with: avg(see page 388), count(see page 388), sum(see page 392) (together with #preceding).

#beforeChildren

See #afterChildren(see page 393).

#children

Only process direct children of the current row.

Example

Summary	X	JOIN#children{X}	SUM#children{X}
T1	3	2, ?	2
T1.1	2		
T1.2		1	1
T1.2.1	1		

```
JOIN#children{X}
SUM#children{X}
```

Can be used with: array(see page 388), sum(see page 392), count(see page 388), avg(see page 388), join(see page 389), min(see page 390), max(see page 389), median(see page 390), percentile(see page 391), quartile1(see page 391), quartile3(see page 391), values(see page 392).

#compact

Ignores undefined values when collecting an array.

This modifier is implicitly turned on by applying the #distinct(see page 395) modifier to the join(see page 389) aggregate function.

Can be used with: array(see page 388).

#depth

Same as #level(see page 397) modifier.

#distinct

#distinct with ARRAY

Makes array(see page 388) only produce an array of distinct values. A duplicate value won't be added more than once if this modifier is on.

Example

Summary	X	ARRAY{X}	ARRAY#distinct{X}
T1	1	1, 2, 2, 3, 1	1, 2, 3
T1.1	2	2	2
T1.2	2	2, 3, 1	2, 3, 1
T1.2.1	3	3	3
T1.2.2	1	1	1

```
ARRAY#distinct{X}
```

#distinct with JOIN

Makes join(see page 389) only concatenate distinct values. A duplicate value won't be added more than once if this modifier is on. When used with arrays, removes undefined values and performs one-step flattening.

Modifiers #beforeChildren(see page 394) and #afterChildren(see page 393) don't work when this option is on.

Example

Summary	X	JOIN{X}	JOIN#distinct{X}	JOIN#subtree#distinct{X}	JOIN#subtree{X}
T1	1	1	1	1, 2, 3	1(1(1, 2, 3), 2, 2)
T1.1	1	1, 1	1	1, 2, 3	1(1, 2, 3)
T1.1.1	1	1, 1, 1	1	1	1
T1.1.2	2	1, 1, 2	1, 2	2	2
T1.1.3	3	1, 1, 3	1, 3	3	3

```
JOIN#distinct{X}
JOIN#subtree#distinct{X}
```

Summary	array	JOIN{array}	JOIN#distinct{array}
T1	1, 2, 2, 3, 1	1, 2, 2, 3, 1	1, 2, 3
T1.1	2	1, 2, 2, 3, 1, 2	1, 2, 3
T1.2	2, 3, 1	1, 2, 2, 3, 1, 2, 3, 1	1, 2, 3
T1.2.1	3	1, 2, 2, 3, 1, 2, 3, 1, 3	1, 2, 3
T1.2.2	1	1, 2, 2, 3, 1, 2, 3, 1, 1	1, 2, 3

```
JOIN#distinct{array}
```

Can be used with: join(see page 389).

#flatten

When collecting values from sub-items (or another subset of related items) if a value is an array, includes all the elements instead of just including the array.

This modifier is implicitly turned on by applying the #distinct(see page 395) modifier to the join(see page 389) aggregate function.

Can be used with: array(see page 388).

#fromDepth

Same as #fromLevel(see page 396).

#fromLevel

Specifies the position of the first row the aggregate function should take as input for a sequence.

Position is specified by an integer parameter denoted as n below:

- Positive values mean the absolute depth of the row in the structure, e.g. n=1 means root.
- Negative values mean the depth relative to current row, e.g. n=-1 is the current item's direct parent.
- Default is 1.

- Zero means the "super-root" row, which is a fictional parent of all the top rows. It can be used to get the value of another aggregate function applied to the whole structure. For example, `JOIN#fromLevel=0{MIN{due_date}}` will provide a sequence of the earliest due dates, starting from the earliest throughout the whole structure, then the earliest throughout the root tree this item is in, and so on.

This modifier does not work with any tree types except #ancestors(see page 394).

Example

Summary	X	JOIN#fromDepth=-1{X}	JOIN#fromDepth=2{X}
T1	1	1	
T2	2	1, 2	2
T3		2, ?	2, ?
T4	4	?, 4	2, ?, 4

```
JOIN#fromLevel=-1{X}
JOIN#fromLevel=2 {X}
```

Can be used with: array(see page 388), join(see page 389), median(see page 390), percentile(see page 391), quartile1(see page 391), quartile3(see page 391), values(see page 392).

#leaves

Only process leaves (items without children) in the subtree of the current row.

Example

Summary	X	JOIN#leaves{X}	SUM#leaves{X}
T1	3	2, 1	3
T1.1	2	2	2
T1.2		1	1
T1.2.1	1	1	1

```
JOIN#leaves{X}
SUM#leaves{X}
```

Can be used with: array(see page 388), sum(see page 392), count(see page 388), avg(see page 388), join(see page 389), min(see page 390), max(see page 389), median(see page 390), percentile(see page 391), quartile1(see page 391), quartile3(see page 391), values(see page 392).

#level

When used with PARENT(see page 390), specifies the position of the parent that possesses value.

Position is specified by an integer parameter denoted as n below:

- Positive values mean the absolute depth of the row in the structure: n=1 means root element, n=2 means an element at the 2nd level, etc.
- Negative values mean the depth relative to the current row: n=-1 is the current item's direct parent.
- Default is -1.
- Zero means the "super-root" row, which is a fictional parent of all the top rows. It can be used to get the value of another aggregate function applied to the whole structure. For example, `PARENT#level=0{SUM{story_points}}` means total story points for the whole structure (including subtrees for all roots).

When used with SUM(see page 392) with the `#preceding` modifier, this specifies the level at which the values should be aggregated.

Example

Summary	X	-1	-2	1	2
T1	3			3	
T1.1	2	3		3	2
T1.2		3		3	
T1.2.1	1		3	3	

```
PARENT#level=-1{X}   // default one
PARENT#level=-2{X}   // "grandparent"
PARENT#level=1 {X}   // root row
PARENT#level=2 {X}
```

Can be used with: parent(see page 390), sum(see page 392) (together with #preceding), count(see page 388) (together with #preceding), avg(see page 388) (together with #preceding).

#levels

Can be used with SUM(see page 392) with the `#preceding` modifier to specify at which levels should the accrual of the values happen.

- It can be a single numeric value, for example: `SUM#preceding#levels=1{story_points}` will accumulate Story Points from top to bottom at level 1.
- It can be a list of numbers, in which case the list must be wrapped in quotes: `SUM#preceding#levels="2,4"{time_spent}` will accumulate Time Spent on levels 2 and 4.
- By default, all levels are counted.

Note that if you use the #baseLevel(see page 394) modifier, only values at levels that are deeper than the base level will be counted.

You can also use #level(see page 397) instead of #levels.

Can be used with: avg(see page 388), count(see page 388), sum(see page 392) (together with #preceding).

#preceding

Can be used with SUM(see page 392) to calculate a numeric total of the current item and all items above it in the structure.

Can be combined with the following modifiers:

- #baseLevel(see page 394) - the sum restarts whenever the specified level is reached

- #levels(see page 398) - the sum will only include the levels specified
- #all(see page 393) - items that appear more than once will be counted multiple times

Example

Summary	X	SUM#preceding{X}	SUM#preceding#baseLevel=1{X}	SUM#preceding#levels="1,3"{X}
T1	1	1		1
T1.1	2	3	2	
T1.2	3	6	5	
T1.2.1	4	10	9	5
T1.2.2	3	13	12	8
T1.3	2	15	14	
T2	2	17		10
T2.1	3	20	3	
T2.2	2	22	5	

```
SUM#preceding{X}
SUM#preceding#baseLevel=1{X}
SUM#preceding#levels="1,3"{X}
```

Can be used with: sum(see page 392).

#reverse

Reverses the order of row processing.

Example

Summary	X	JOIN#reverse{X}
T1	3	3
T1.1	2	2, 3
T1.2		?, 3
T1.2.1	1	1, ?, 3

```
JOIN#reverse{X}
```

Can be used with: array(see page 388), join(see page 389).

#separator

Defines the separator for string joining. This modifier has a string parameter. The default is ", ".

Example

Summary	X	JOIN#separator="->"{X}
T1	3	3
T1.1	2	3->2
T1.2		3->?
T1.2.1	1	3->?->1

```
JOIN#separator="->"{X}
```

Can be used with: join(see page 389).

#strict

Do not process the current row item as part of the aggregation.

Cannot be used together with #children(see page 394), #ancestors(see page 394) or #leaves(see page 397), since these already exclude the current row.

Example

Summary	X	JOIN#strict{X}	SUM#strict{X}
T1	3	2, ?(1)	3
T1.1	2		
T1.2		1	1
T1.2.1	1		

```
JOIN#strict{X}
SUM#strict{X}
```

Can be used with: array(see page 388), sum(see page 392), count(see page 388), avg(see page 388), join(see page 389), min(see page 390), max(see page 389), median(see page 390), percentile(see page 391), quartile1(see page 391), quartile3(see page 391), values(see page 392).

#subtree

Process the whole subtree of the current row. This is the default behavior for sum(see page 392), count(see page 388), avg(see page 388), min(see page 390), max(see page 389).

Example

Summary	X	JOIN#subtree{X}	SUM{X}
T1	3	3(2, ?(1))	6
T1.1	2	2	2
T1.2		?(1)	1
T1.2.1	1	1	1

```
JOIN#subtree{X}
```

Can be used with: array(see page 388), sum(see page 392), count(see page 388), avg(see page 388), join(see page 389), min(see page 390), max(see page 389), median(see page 390), percentile(see page 391), quartile1(see page 391), quartile3(see page 391), values(see page 392).

#toDepth

Same as #toLevel(see page 401).

#toLevel

Specifies the position of the last row the aggregate function should take as input for a sequence.

Position is specified by an integer parameter denoted as n below:

- Positive values mean the absolute depth of row in the structure, e.g. n=1 means root.
- Negative values mean the depth relative to current row, e.g. n=-1 is the current item's direct parent.
- 0 means current row.
- Default is 0.

This modifier does not work with any tree types except #ancestors(see page 394).

Example

Summary	X	JOIN#toDepth=-1{X}	JOIN#toDepth=2{X}
T1	1		
T2	2	1	1, 2
T3		1, 2	1, 2
T4	4	1, 2, ?	1, 2

```
JOIN#toLevel=-1{X}
JOIN#toLevel=2 {X}
```

Can be used with: array(see page 388), join(see page 389), median(see page 390), percentile(see page 391), quartile1(see page 391), quartile3(see page 391), values(see page 392).

#truthy

Only count row if the subexpression produces a truthy value (see page 0).

Example

Summary	X	COUNT#truthy{X}
T1	0	2
T1.1	2	1
T1.2		1
T1.2.1	1	1

```
COUNT#truthy{X}
```

Can be used with: count(see page 388).

3.8.3.4 Standard Variable Reference

As you write your formula, Structure attempts to map your variables to well-known attributes:

- Variables with the same name as an attribute are mapped to those attributes. For multi-word attributes, you can write the words without spaces, or place a _ between each word: `affectsversion` or `affects_version`.
- Many commonly-used variations of attribute names are mapped to the corresponding attribute.
- In addition to Jira fields, some variable names are automatically mapped to common Structure attributes. See Structure Attributes(see page 405).

When naming variables:

- Do not include spaces. Omit spaces or use an underscore.
- Case is ignored (AffectsVersion is the same as affectsversion)

Jira Fields

The following list contains the most commonly-used variables that are automatically be linked to a Jira field. This is not a complete list.

Jira Field	Standard Variable Name	Type	Notes
Affects Version/s	`affectsVersions`	Array<Version (see page 412) >	
Assignee	`assignee`	User(see page 408)	
Comments	`comments`	Array<Comment (see page 409) >	
Components	`components`	Array<Component (see page 409) >	

Jira Field	Standard Variable Name	Type	Notes
Created	`created`	Number (Date/Time)	
Creator	`creator`	User(see page 408)	
Description	`description`	Text	
Due Date	`dueDate`	Number (Date)	
Environment	`environment`	Text	
Epic Link	`epic`	Issue(see page 406)	Can also use: epicLink
Epic Stories	`epicStories`	Array<Issue(see page 406)>	For epics, list issues belonging to the epic. For non-epics, undefined. Can also use: stories
Fix Version/s	`fixVersions`	Array<Version(see page 412)>	Can also use: versions
Key	`key`	Text	Can also use: issueKey
Issue ID	`issueid`	Number	
Issue Links	`issueLinks`	Array<Issue Link(see page 410)>	
Issue Type	`issueType`	Issue Type(see page 408)	
Labels	`labels`	Array<Label>	Label items have no properties. Item's text representation is a label itself.
Original Estimate	`originalEstimate`	Number (Duration)	

Jira Field	Standard Variable Name	Type	Notes
Parent Issue	`parentTask`	Issue(see page 406)	
Priority	`priority`	Priority(see page 410)	
Project	`project`	Project(see page 411)	
Remaining Estimate	`remainingEstimate`	Number (Duration)	
Reporter	`reporter`	User(see page 408)	
Resolution	`resolution`	Resolution(see page 411)	
Resolved	`resolved`	Number (Date/Time)	Can also use: resolutionDate
Sprint	`sprint`	Sprint(see page 411)	
Sprint End Date	`sprint_end_date`	Number (Date/Time)	
Sprint Goal	`sprint_goal`	Text	
Sprint Start Date	`sprint_start_date`	Number (Date/Time)	
Sub-Tasks	`subtasks`	Array<Issue(see page 406)>	
Status	`status`	Status(see page 412)	
Status Category	`statusCategory`	Text	
Summary	`summary`	Text	
Time Spent	`timeSpent`	Number (Duration)	

Jira Field	Standard Variable Name	Type	Notes
Updated	`updated`	Number (Date/Time)	
Version Description	`version_description`	Text	
Version Start Date	`version_start_date`	Number (Date/Time)	
Version Release Date	`version_release_date`	Number (Date/Time)	
Votes	`votes`	Number	
Watchers	`watchers`	Number	
Work Logs	`workLogs`	Array<Work Log(see page 413)>	

Structure Attributes

The following variables are automatically mapped to Structure attributes.

Variable Name	Type	Results

`item`
 	
`this`
	*	Returns the value of type "Item" representing the item for which the formula is calculated.
		
		If the formula is calculated for an issue, it will be of type Issue(see page 406).
		
		If, for example, you group issues by Sprint, you will have rows in the structure representing sprints. For those rows, the "item" variable will return items of type Sprint(see page 411).

`itemid`	Text	Returns the item type plus the item ID. Ex. "issue/10800"
`itemType`	Text	Returns the item type: issue, project, user, folder, etc.
`level`	Number (Integer)	Returns the level of the item within the structure's hierarchy. Can also use: depth

Variable Name	Type	Results
`notes`	Text	Returns the text entered into Notes column
`done`	Number (Boolean)	Returns a '1' if the issue is Done; otherwise, '0'
`editable`	Number (Boolean)	Returns a '1' if the item can be edited; otherwise, '0'

`sum_`
 	
`total_`
	Number	Either prefix, attached to a well-known variable name afterwards, such as **Sum_Story_Points** or **Total_Estimate**,
		
		are converted to a Sum(see page 387) attribute of the given value (without the duplicate removal option)

3.8.3.5 Item Property Reference

Expr allows you to access item properties within formulas, using the following notation: `item.property`. The available properties depend on the item type.

See below for a complete list of item types and their supported properties.

Issue

Accessible via: *item, subtasks, parent, epic, epicStories*

Property Name	Type	Comments
`key`	Text	

Property Name	Type	Comments
summary	Text	
description	Text	If wiki markup is used, the value is the marked up text (not HTML)
environment	Text	
created	Number (Date/Time)	
updated	Number (Date/Time)	
resolutionDate	Number (Date/Time)	
isSubtask	Number (Boolean)	
votes	Number	
watches	Number	
dueDate	Number (Date)	
timeSpent	Number (Duration)	
originalEstimate	Number (Duration)	
remainingEstimate	Number (Duration)	
url	Text	Issue's URL
subtasks	Array<Issue>	
parent	Issue	Parent issue, if this is a sub-task
assignee	User	
reporter	User	
creator	User	
issueType	IssueType	
priority	Priority	
project	Project	
resolution	Resolution	
status	Status	

Property Name	Type	Comments
`affectedVersions`	Array<Version>	
`fixVersions`	Array<Version>	
`components`	Array<Component>	
`attachments`	Array<Attachment>	
`labels`	Array<Text>	
`customField_10011`	Variable	Custom field value for custom field identified by its ID.
`customFieldName`	Variable	Custom field value for the field identified by its name.

IssueType

Accessible via: *issueType*

Property Name	Type	Comments
`name`	Text	
`description`	Text	
`sequence`	Number	Used to sort issue types.
`isSubtask`	Number (Boolean)	True if the type is a sub-task type.

User

Accessible via: *assignee, reporter, creator*

Property Name	Type	Comments
`name`	Text	User's login name
`key`	Text	User key (used internally by Jira)
`displayName`	Text	User's full name

Property Name	Type	Comments
`emailAddress`	Text	User's email (if available)
`isActive`	Number (Boolean)	True if the user is active

Attachment

Accessible via: *attachments*

Property Name	Type	Comments
`created`	Number (Date/Time)	
`fileSize`	Number	
`fileName`	Text	
`author`	User	

Comment

Accessible via: *comments*

Property Name	Type	Comments
`author`	User	
`updateAuthor`	User	
`created`	Number (Date/Time)	
`updated`	Number (Date/Time)	
`body`	Text	

Component

Accessible via: *components*

Property Name	Type	Comments
`name`	Text	
`description`	Text	
`lead`	User	

Issue Link

Accessible via: *issueLinks*

Property Name	Type	Comments
source	Issue	
destination	Issue	
sequence	Number	
type	IssueLinkType	

Issue Link Type

Accessible via: *issueLinks.type*

Property Name	Type	Comments
name	Text	
inward	Text	
outward	Text	
style	Text	
isSubTask	Number (Boolean)	
system	Number (Boolean)	

Option

Accessible via: *"select" custom fields*

Property Name	Type	Comments
isDisabled	Number (Boolean)	
value	Text	
parent	Option	Useful with cascading select fields
sequence	Number	

Priority

Accessible via: *priority*

Property Name	Type	Comments
name	Text	
description	Text	
sequence	Number	

Project

Accessible via: *project*

Property Name	Type	Comments
name	Text	
key	Text	
url	Text	
email	Text	
lead	User	
description	Text	

Resolution

Accessible via: *resolution*

Property Name	Type	Comments
name	Text	
description	Text	
sequence	Number	

Sprint

Accessible via: *sprint*

Property Name	Type	Comments
name	Text	
startDate	Number (Date/Time)	

Property Name	Type	Comments
endDate	Number (Date/Time)	
completeDate	Number (Date/Time)	
state	Text	
sequence	Number	
goal	Text	

Status

Accessible via: *status*

Property Name	Type	Comments
name	Text	
description	Text	
sequence	Number	
category	Text	

Version

Accessible via: *affectsVersion, fixVersion*

Property Name	Type	Comments
name	Text	
description	Text	
project	Project	
isArchived	Number (Boolean)	
isReleased	Number (Boolean)	
releaseDate	Number (Date)	
startDate	Number (Date)	

Work Log

Accessible via: *workLogs*

Property Name	Type	Comments
`author`	User	
`updateAuthor`	User	
`created`	Number (Date/Time)	
`updated`	Number (Date/Time)	
`comment`	Text	
`startDate`	Number (Date/Time)	
`timeSpent`	Number (Duration)	

Change History Group

Accessible via: *history*

Property Name	Type	Comments
author	User	
time	Number (Date/Time)	
timestamp	Number (Date/Time)	Alias for time
timePerformed	Number (Date/Time)	Alias for time
issue	Issue	
changedIssue	Issue	Alias for issue
changes	Array<Change History Item>	
changeItems	Array<Change History Item>	Alias for changes

Change History Item

Accessible via: *history.changes*

Property Name	Type	Comments
`field`	Text	The name or id of the changed field. Can be used in ACCESS and HISTORICAL_VALUE functions.

Property Name	Type	Comments
`from`	Variable	Previous value, which corresponds to the field's type. May be a text value if the value of the field type is impossible to retrieve.
`fromText`	Text	Text representation of the previous value. This can sometimes be faster than using "from".
`to`	Variable	Changed value, which corresponds to field's type. May be a text value if the value of the field type is impossible to retrieve.
`toText`	Text	Text representation of the changed value.This can sometimes be faster than using "to".
`incremental`	Number (Boolean)	When true, the field's "from" and "to" indicate a change of a single value in a multi-valued field. (Add, remove, or change)
`changeGroup`	Change History Group	The Change History Group parent entity. Can be used to access the time and author of the change item.

3.8.3.6 Expr Error Codes

Evaluating Expr expressions may produce errors. Normally these errors are shown to the user with a human-readable message. However, in some cases you might need to check for a specific error using the ISERR(see page 0) function.

Error Code	Name	Displayed As	Description	How to Fix
1	Parse Error	???	The expression is invalid. The formula editor should highlight the problematic place with red. If you hover the mouse pointer over the red area, a tooltip with the details of the problem is shown.	Review and edit the expression.

Error Code	Name	Displayed As	Description	How to Fix
2	Unknown Function	FUNC?	The expression calls a function that is not available or does not exist. This error is also shown if you call a user-defined function, but the variable is actually not of the user function type, for example: `WITH x = 1 : x(1)`.	1. Review used functions in the expression to see if there are any typos in the names. Check the Expr Function Reference(see page 345). 2. If you're using user-defined functions, make sure the names you're calling refer to functions.
3	Bad Number of Arguments	ARGS?	A function is used with an incorrect number of arguments.	Review the expression and see if all functions are called with a correct number of arguments. Check the Expr Function Reference(see page 345).
4	Arithmetic Error	DIV/0	An arithmetic error was encountered. Most often it is division by zero, but it may also be something else, such as passing a non-integer value to a function that expects only an integer.	Review your formula to ensure you're not dividing by zero. (To avoid division by zero, use the IF expression.) If this does not work, try separately calculating parts of the formula to identify the error.
5	Variable Error	VAR!	An attribute that a variable was bound to produced an error.	To fix, review the attributes bound to the expression variables and see why they could have produced an error for the row that shows this error. Try viewing the related attributes in separate (temporary) columns.
6	Function Execution Error	FUNC!	A function suffered an internal error.	Contact your Jira administrator so they can take a look at the log files.

Error Code	Name	Displayed As	Description	How to Fix
				Contact Tempo Support[61] if possible. As a workaround, review the latest changes you made to the formula and try to achieve the same some other way.
7	Value Type Error	TYPE?	A value is of the wrong type and could not be converted to the type required by a function. This error happens, for example, if a text value cannot be converted to a number, or if a fractional number is used where an integer is required.	Check the formula and function calls, make sure the values passed to the functions are of expected type. Evaluate parts of the expression to debug.
8	Malformed Regex	REGEX?	A text with invalid regular expression was passed to a matching function.	Check the regular expressions that you use. See Expr Pattern Matching(see page 417) for details.
9	Internal Error	ERROR!	There's an internal problem with Jira or Structure.	Contact your Jira administrator so they can take a look at the log files. Contact Tempo Support[62] if possible.
10	Invalid Value	VALUE!	An invalid value was passed as a function argument. For example, a string that could not be parsed as a date.	Check arguments for the functions that expect specific values.
11	Aggregation Error	AGGR?	The formula contains an unknown aggregate function or an invalid aggregate function modifier.	Check that your aggregate function and modifiers(see page 387) are valid and spelled correctly.

61 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
62 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Error Code	Name	Displayed As	Description	How to Fix
12	Execution Limit	LIMIT!	The formula is too complex or caused deep recursion. This error happens when a formula requires too many calculation steps, or if it contains too many nested function calls.	If the formula is indeed too large, consider breaking it down into multiple parts and calculating each part in a separate column. If the formula does not seem too large, see if it applies a user function to a very large data set. Consider ways to reduce the data set. If the formula is sophisticated, check if there may be endless recursion. Finally, you can contact your Jira administrator and discuss raising the execution limits via Advanced Configuration and Dark Features(see page 974).
13	Array Too Large	ARRAY!	The array is too large. Structure places a limit on how many values an array may have.	Consider reasons for having so many values in one array. Perhaps you could use an aggregation function instead.
14	Text Too Long	TEXT!	A text value is too long. Structure places a limit on how many characters a text value may contain.	Consider reasons for having such a long text value. Does your formula use the REPEAT function or convert long arrays to text?

3.8.3.7 Expr Pattern Matching

Expr language allows you to check text values against a certain pattern, when using the MATCH(see page 0), CASE(see page 0), REPLACE(see page 0) or SEARCH(see page 0) function.

There are three types of patterns that can be used: Exact Matching(see page 418), Wildcard Matching(see page 418), and Regular Expression Matching(see page 418).

Exact Matching

This is the simplest pattern type, which compares `value` against an exact text value:

- `MATCH(value, "Apples")`

Although it's called "exact matching", there are some additional rules that make the matching easier:

- All leading and trailing whitespace characters are removed from the value.
- Text comparison is case-insensitive, which means `APPLES` will match `Apples`.
- The value (without leading and trailing spaces) must match the whole pattern.

Exact matching is used by default, unless the pattern is recognized as requiring Wildcard or Regular Expression matching.

Wildcard Matching

Wildcard patterns let you use the wildcard symbol "`*`" to specify any number of any characters (including no characters).

- `MATCH(value, "App*")`

The above function would return "1" for any value that started with the characters "App" – so "App", "Apple" and "Apples are good for you" would all match. You can also use multiple asterisks to build your pattern. `Match(value, "A*L*")` would match anything that starts with an A and contains an L, including "Apples", "Almanac" and "Aunt Sal".

Wildcard matching uses the same rules as exact matching:

- All leading and trailing whitespace characters are removed from the value.
- Text comparison is case-insensitive, which means `APPLES` will match `App*`.
- The value (without leading and trailing spaces) must match the whole pattern.

Wildcard matching is used when the pattern is not recognized to be a Regular Expression Pattern but contains at least one asterisk.

Regular Expression Matching

This type of matching lets you use powerful regular expressions to specify exactly what you need to match with.

- `MATCH(value, "/^Ap+.*s$/")`

Structure uses regular expressions available with Java. For full documentation about the regular expression language, see Java documentation for Pattern[63].

The regular expression matching is different from other types of matching. The following rules apply:

- Leading and trailing whitespace characters are **not** removed.
- Text comparison is case-insensitive, like with the other types of matching.
- The value does not have to fully match the pattern – it is sufficient that at least one occurrence of the pattern is found in the value. To make your pattern match the whole text, use "^" and "$" characters in the pattern.

Regular expression matching is turned on if the first and the last characters of the pattern are "`/`". (These characters are removed, as they are not a part of the pattern.)

63 https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

3.8.3.8 Expr validation

The Expr language does not allow certain constructs in a formula. When a user creates a formula with one of these, the **???** error is shown, and the disallowed section of formula is highlighted in red.

The following situations will cause validation errors:

- **Illegal user function name** - A user function name shouldn't coincide with a system function name.
- **Same parameter names in user function** - All parameter names for a single user function must be unique.
- **Local variable usage in aggregations** - A local variable cannot be used in an aggregation function, if the variable is declared outside of the aggregation function. Local variables declared inside curly braces are allowed. The same is applicable to user function parameters and the implicit parameter "$".
- **"$" sign is used without a proper context** - "$" can only be used as a part of the second argument of invocation, and only for functions FILTER, MAP, REDUCE, GROUP, ANY, ALL, NONE, SORT_BY, UMIN_BY, UMAX_BY.
- **Unsupported aggregation modifier** - If a modifier isn't recognized by an aggregation, it's an error. This only affects modifier names.
- **Wrong aggregation modifier value** - Some aggregation modifiers require a value to be passed in the right format. For example, specifying parent#level=1.5 is an error.
- **Error in embedded JQL or S-JQL query** - JQL and S-JQL have their own syntax and may produce errors if a query is not correct. These errors are highlighted in the same manner as other errors.

3.8.3.9 Work Time in Formula Columns

When the format of a formula column(see page 229) is set to **Work Time**, Structure uses Jira's time tracking settings to convert the number of hours to the number of days and weeks. By default, Jira is configured for an 8 hour work day, with 5 work days per week.

> ⓘ If Duration is selected, but Work Time is not checked, hours are converted to days and weeks on a calendar basis.

Whether you need to use the **Work Time** option depends on where the value is coming from:

- When working with specific dates, you will probably want to keep the Work Time option off and see the calendar duration. For example, if you want to calculate the number of days a ticket remains open (`now() - created`), leaving the Work Time option will give you a more accurate result.
- When working with values retrieved from an issue's Original Estimate, Remaining Estimate and Time Spent fields, you will probably want to use Work Time option. For example, to calculate overspending (`time_spent + remaining_estimate - original_estimate`), selecting the Work Time option will give a result based on actual work hours.

3.8.4 Changes to Expr in Structure 7

Structure 7.0 brings major improvements to the Expr language, making it an order of magnitude more powerful. This page lists the main additions to the language and the function library and is intended for the users of Structure 6 and earlier, already familiar with Expr(see page 235).

3.8.4.1 New Features

Arrays

Expr now supports arrays, or lists of values. For example, the Fix Versions field may contain multiple values - in Structure 7.0, there are functions that pull out specific values from these fields, does an operation to every value, and more!

We've also added several new functions to work with arrays: Array Functions(see page 423)

Properties

Formulas can now get the value of a particular property of an item using the following notation: `object.property`

```
fixVersion.releaseDate
```

This returns the release date for the fixVersion.

You can also string multiple property calls together:

```
project.lead.email
```

This returns the email address of the lead for the project.

For a complete list of supported properties, see Item Property Reference(see page 406).

Chained Function Calls

This new way to call a function allows you to conveniently apply a sequence of functions to a value by listing each function one after the other, separated by a (.) dot.

- Old method: `F3(F2(F1(x)))`
- New method: `x.F1().F2().F3()`

With this new method, the value that comes before the dot becomes the first argument for the function. If the function takes multiple arguments, the rest of the arguments must be written in the parentheses.

For example:

```
created.FORMAT_DATETIME("yyyy").CONCAT(" year issue")
```

If the issue was created in 2021, the result would be: "2021 year issue"

The chain method can be applied to any expression and any function - including User Functions and Arrays: `ARRAY(1, 2, 3).MAX().SQR()`

User Functions

A User Function allows you to define a locally-used function within a formula. User functions can be defined in a similar manner as local variables:

```
WITH square(x) = x * x :
  square(impactField) / square(storyPoints)
```

In this example, the user function is given a name ("square") and then used to perform the same calculation on multiple fields. To learn more, see the language reference(see page 0).

User Functions for Arrays - introducing the "$" character

When you need to perform an operation on each element in an array, you can use “$” to indicate each element in the array.

For example, if you want to filter work logs based on whether the author is the current user, you could write:

```
worklogs.FILTER($.author = ME())
```

In this formula, Structure replaces "$" with each element of the array and calculates the value of "$.author = ME()" expression - if the author is the current user, it returns true and that work log will be included in the FILTER results.

This method becomes very powerful when you combine multiple user functions together. To learn more, see the language reference(see page 316).

Embedded Queries

It is now possible to include JQL and S-JQL (our own Structured JQL(see page 428)) within a formula:

```
JQL { assignee = currentUser() }

SJQL { leaf and child of [ assignee = currentUser() ] }
```

The result will be a boolean value – true (number 1) if the issue (that the formula is calculated for) satisfies the query.

Since JQL is a Jira-based query, it will work only on issues; the result will be false (`0`) on other types of items. S-JQL can be used for more complex queries applicable to the whole structure.

For example:

```
// Collect total story points from all sub-issues assigned to members of Team2 group,
unless the stories are under folder "Special"
SUM {
    IF JQL { assignee in membersOf("Team2") } :
    IF NOT SJQL { descendant of folder("Special") } :
       storyPoints
}
```

Enhanced IF Expression

It is now possible to create complex IF statements, as well as IF/ELSE statements within a formula.

```
WITH total = x + y :
  IF total > 0 :
     x / total
  ELSE : error
```

Note: the ":" after "ELSE" is optional – in the example above, we've included it for readability.

Concatenation Operator

The operator CONCAT allows you to join two text strings together:

```
  issueLink.source
  CONCAT
  " depends on "
  CONCAT
  issueLink.destination
```

If the link.source is issue PR-111 and the link.destination is PR-231, the above code would return: `PR-111 depends on PR-231`

Text Snippets

Text Snippets allow you to generate texts using variables and expression. This is particularly helpful in formulas that utilize wiki markup.

When using text snippets:

- The snippet should be enclosed with `"""`
- It may be multi-line
- **`$var`** is replaced with the value of a variable var
- **`${expression}`** is replaced with the value of the expression

```
""" $var1 + $var2 = ${var1 + var2} """

""" this $glass is half-${
IF optimist:
   'full' ELSE: 'empty'} """
```

3.8.4.2 New Functions

The following functions have been added or updated in Structure 7.0. For details about each function and how to use them, see Expr Function Reference(see page 345).

Array Functions

- ARRAY - Creates an array from a list of elements.
- GET - Retrieves an element from an array based on its index. Array indexes are 0-based. Returns undefined if index is out of array bounds.
- UNIQUE - Removes duplicates from the array. The order of non-duplicate elements is preserved.
- COMPACT - Removes all undefined values from the array.
- FLATTEN - Given an array of arrays, makes a one-step flattening.
- RECURSIVE_FLATTEN - Performs recursive flattening and compacting of the array, producing an array guaranteed to be flat and not contain undefined values.
- REVERSE - Reverses the order of elements in the array.
- FILTER - Filters the array so that it retains only elements for which a specified user function returns a truthy value[64].
- MAP - Applies a user function to every element of the array.
- SORT - Produces a sorted array.
- SORT_BY - Sorts the array by comparing the values produced by calling a user function for each array element.
- REDUCE - Reduces an array to a single value.
- GROUP - Groups array elements into buckets based on the value produced by a user function.
- JOIN - Produces a text representing the array by converting each element to text and joining them together.
- CONTAINS - Returns true (1) if the array contains a specified element.
- ANY - Checks if any of the array elements satisfies a condition.
- ALL - Checks if all of the array elements satisfy a condition.
- NONE - Checks that none of the array elements satisfy a condition.
- SEQUENCE - Creates an array of sequential integer numbers.
- INDEXES - Creates an array of indexes of another array.
- SIZE - Returns the number of elements in an array.
- MERGE_ARRAYS - Produces a single array with the elements of all parameter arrays.
- SUM - Produces a total of all numeric elements in the array.
- MUL - Produces the product of all numeric elements in the array.
- SUBARRAY - Produces an array with some elements from the given array.
- CONTAINS_ALL - Checks if one array contains all elements of another array.
- CONTAINS_ANY - Checks if one array contains at least one element from another array.
- INDEX_OF / LAST_INDEX_OF - Returns the index of a first/last occurrence of an element in an array.
- FIRST / LAST - Returns first/last element of the array.
- WITHOUT - Returns the input array with all elements except those equal to a specified value.
- IS_EMPTY - Checks if array is empty.

64 https://wiki.almworks.com/display/structure2gmaster/Expr+Language+Reference#ExprLanguageReference-ConversiontoBoolean:FalsyandTruthyValues

Statistical Functions

- MIN/MAX - When used with multiple arguments, MIN/MAX find the smallest / largest value among the numbers passed as parameters. When used with an array, finds the minimum/maximum number in the array.
- UMIN/UMAX - Same as MIN/MAX, but does a universal comparison, accepting any type of values, including entities.
- UMIN_BY/UMAX_BY - Returns the minimum/maximum by comparing them using values calculated by calling a user function for each element.
- PERCENTILE - Calculates Nth percentile of the values in the given array.
- MEDIAN - Calculates a median value of an array of numbers.
- QUARTILE - Calculates a quartile value of an array of numbers.
- AVERAGE - Calculates an average of the numbers in the array.
- STDEV/STDEVP - Calculates standard deviation (based on a sample or on entire population).

Other Functions

- URL_ENCODE - Translates a text into application/x-www-form-urlencoded format.
- URL_DECODE - Decodes an application/x-www-form-urlencoded text.
- SPLIT - Produces an array from a text by splitting it using a separator.
- ACCESS - Tries to access a property of an item.

Aggregate Functions

The following Aggregate functions have been added to work with the new features and functions introduced in Structure 7.0. See Aggregate Function Reference(see page 387) for more detail.

- ARRAY { } - Calculates an expression for each sub-issue (or other set of rows as defined by the modifiers), and collects the values in an array.
- VALUES { } - Same as ARRAY { }, but removes duplicates, undefined values, and expands (flattens) the array values produced by the inner expression. For example, **`VALUES { fixVersion }`** would produce a list of all fix versions set for the sub-issues.
- MEDIAN { }, PERCENTILE { }, and other - calculates a numerical expression for each sub-issue (or another set of issues), and then calculates the corresponding statistical value.

Changes and Notes for Existing Aggregate Functions

The following changes have been made to the existing aggregate functions because in Structure 7.0 the inner expression (the one inside curly braces) can now produce new types – arrays, items, key-value maps and user functions.

Aggregate Function	Function	Change in Structure 7.0	Expr: Array	Expr: Item	Expr: Key-Value Map
SUM	Sums the result of the inner expression for each subject row. If #distinct is used, does not count the same **item** twice.	If the inner expression produces an array, sums elements of that array.	Sums elements of the array.	Error	Error
MIN	Finds a minimum value of expr calculated for each row.	If expr produces an array, finds the minimum element within that array.	Uses minimum value from the array.	Uses items's order.	Sorted as if it was an undefined value
MAX	Finds a maximum value of expr calculated for each row.	If expr produces an array, finds the maximum element within that array.	Uses maximum value from the array.	Uses items's order.	Sorted as if it was an undefined value
JOIN	Creates a string representation of values produced by expr, joined together as a list or hierarchically. If #distinct is used, does not include the same **value** twice.	If expr produces an array, joins elements of that array.	Uses conversion to Text/Joined.	Uses conversion to Text/Joined.	Error

Note: If the inner expression produces a user function, these will always result in an error.

3.8.4.3 Backward Compatibility

Some of the enhancements we've made required us to change the way formulas work with certain items or characters. Because of this, some old formulas may need to be updated, as they will either no longer work with Structure 7, or will cause an unexpected result.

Please review the following chart and update your existing formulas as necessary.

Change	Sample Formula	Result Before Structure 7.0	Result After Structure 7.0
Dot character (".") is no longer a valid character for a variable.	WITH my.data = 5 : ... CONCAT("user", user.name[65]) with user.name[66] bound to an attribute	valid calculation	parse error user.name[67] undefined
TRIM() function on array values will trim each element.	*Assume we have version names with leading and trailing spaces, " v1 " and " v2 ".* TRIM(fixVersion)	"v1 , v2" *(spaces inside)*	"v1, v2" Or ["v1 ", "v2"] when treated as an array
MIN / MAX functions now require specific parameters; otherwise, they can now result in errors.	min(1, "aha!")	1	"VALUE?" error
Third party attribute loaders that provide values in any format except TEXT, DURATION, TIME, NUMBER, BOOLEAN will not be loaded by formula anymore. The possibility to add it via variable selector stands.	x x is bound to a loader with an unsupported format.	text representation of the loaded value	undefined value
Formula errors are now propagated through aggregations	join { 1/0 }	"?"	"DIV/0" error
We introduced execution limits. Formulas that require too many calculation steps will return an error because they will take too long to process	overly-complex formula	valid calculation	"LIMIT!" error

65 http://user.name
66 http://user.name
67 http://user.name

Change	Sample Formula	Result Before Structure 7.0	Result After Structure 7.0
Array size limit may influence variables usage and join#distinct	fixVersions join#subtree#distinct { key }	valid calculation	"ARRAY!" error
Join#distinct functions a little differently. Previously, in the example to the right, it would have interpreted the two sets of labels as "a, b" and "b, c" - now it's able to identify the individual labels within each set ("a" and "b" and "b" and "c").	join#distinct{labels}, labels are (a, b) and (b, c)	a, b, b, c	a, b, c
Previously it was possible to write one field to another using an effector, even if the parameters did not match. For example, you could write a resolution to a status. This is no longer possible, if types do not coincide. Workaround - In some cases, this will still work if you wrap the field to change its type: instead of "resolution" use "text(resolution)".	resolution	attempts to write the value	an error

3.8.5 Comparision Between Formulas in Cloud and Data Center

Users who are familiar with the current versions of Structure for Jira Data Center or Server will notice some differences in the features available on Structure Cloud. We will work to bring many (if not all) of these features to Structure Cloud as soon as possible.

Feature	DC/Server	Cloud
Expr Language	Expr 2.0	⚠️ Expr 2.0 with limitations
Values	➕	⚠️ Item Properties are only available for User, Component, Project, Sprint, Status, and Version types.
Variables	➕	⚠️ Only Jira fields can be used as variables ⚠️ "this" variable is not supported
Local Variables	➕	➕
Conditional Expression	➕	➕

Property Access	➕	⛔
Functions	➕	⚠️ No Miscellaneous Functions: ACCESS, HISTORICAL_VALUE, ME ⚠️ No user functions: USER_NAME, USER_EMAIL, USER_IS_ACTIVE, USER_DISPLAY_NAME ⚠️ Only our format is supported in DATE, DATETIME functions ➕ yyyy-MM-dd for DATE, yyyy-MM-dd HH:mm:ss for DATETIME ⛔ Standard formats for the specified locale are not supported ⛔ Jira formats, as specified in the Jira's system settings are not supported
User-defined Functions	➕	➕
Aggregate Functions	➕	➕
Embedded Queries	➕	⚠️ Only JQL, no S-JQL
Other		
Attribute sensitivity	➕	⚠️ Standard sensitivity. All attribute fields except numeric and date fields are set to sensitive. Learn more(see page 960)
Jira Duration	➕	➕
Wiki Markup	➕	⚠️ Supports Markdown languge[68] for text formatting, color, and emojis.

3.9 Structured JQL

Structure not only displays a hierarchy of items, it also allows you to search items based on their relative positions in the hierarchy. The language used to express such queries is called **Structured JQL** or **S-JQL** (where JQL stands for Jira Query Language).

Structured JQL queries are available in these places:

- inside a `structure()` JQL function — this allows you to search for issues in a structure on the Issue Navigator page — see structure() JQL function(see page 455)
- in the Search Area on the Structure Widget — see Search(see page 205)
- in the S-JQL Filter generator(see page 158)
- in the workflow validator or condition — see Workflow Integration(see page 969)

68 https://wiki.almworks.com/display/structure/.Markdown+in+Formula+Columns+v8.3

3.9.1 Get Started with S-JQL

We recommend starting with the S-JQL Cookbook(see page 429), which contains a number of common examples that can be used as is or modified to meet your specific needs.

3.9.2 Master S-JQL

For a comprehensive description of the language and `structure()` JQL function, consult the S-JQL Reference(see page 434).

- S-JQL Cookbook(see page 429)
- S-JQL Reference(see page 434)
- structure() JQL function(see page 455)

3.9.3 S-JQL Cookbook

Here are the most common examples of using S-JQL.

3.9.3.1 Find issues added to a structure

Goal: Suppose that you are using a structure named "My todo list" as a collection of issues, and you want to see in the Issue Navigator all issues added to this structure.

How to achieve: In the Issue Navigator, switch to Advanced Searching[69] and run the following query:

```
issue in structure("My todo list")
```

If you want to find issues added to the Default Structure(see page 537), you can omit the structure name:

```
issue in structure()
```

69 http://confluence.atlassian.com/display/JIRA/Advanced+Searching

^ up to the list of examples(see page 429)

3.9.3.2 Quick Filter for JIRA Agile's (GreenHopper) Scrum Board to display only low-level issues in a structure

Setup: Suppose that you are using a structure named "Project work breakdown" to organize tasks under higher-level "container" issues that provide an overview of your team's work. In this setting, the actual tasks are at the bottom level of the hierarchy. Also, suppose you are using JIRA Agile's Scrum Board to manage your sprints.

Goal: You want to see only the actual tasks in backlog, hiding the container issues.

How to achieve: Add a Quick Filter[70] to your JIRA Agile (GreenHopper) board with the following JQL:

```
issue in structure("Project work breakdown", leaf)
```

If your structure is organized such that *two* lower levels matter to you on the JIRA Agile board, you'll search for leaf issues and their parents with this JQL:

```
issue in structure("Project work breakdown", "leaf or parent of leaf")
```

^ up to the list of examples(see page 429)

3.9.3.3 Retrieve all Epics in a certain status and all of their children

Setup: You have a structure named "Enterprise Portfolio" with Epics on the top level, Stories beneath them, and Tasks with their Sub-Tasks occupying the lower levels of the hierarchy.

Goal: You need to see Epics in status *Assigned* with all of their children.

How to achieve: In the Issue Navigator, switch to Advanced Searching[71] and run the following query:

```
issue in structure("Enterprise Portfolio", "issueOrAncestor in [type = Epic and
status = Assigned]")
```

If you want to see these issues in the structure, go to Structure Board(see page 88) and type this query in the Search Area(see page 205) in the JQL mode.

Also, you can type only the last part of the query if you use S-JQL search mode(see page 207):

```
issueOrAncestor in [type = Epic and status = Assigned]
```

^ up to the list of examples(see page 429)

70 https://confluence.atlassian.com/display/GH/Configuring+Quick+Filters
71 http://confluence.atlassian.com/display/JIRA/Advanced+Searching

3.9.3.4 Find Test Cases associated with Stories in an active sprint

Setup: Suppose that you have a structure named "Enterprise Portfolio Testing", where you have Epics on the top level, Stories on the second level, then come Test Sub-Tasks, and finally Test Cases.
You are also using JIRA Agile (Greenhopper) to manage your sprints, which contain Stories. The fact that a Test Case is associated with an Story is recorded only in the structure.

Goal: You need to find those Test Cases that are associated with Stories in an active sprint.

How to achieve: You can use Issue Navigator's Advanced Searching[72] capability or open the structure on the Structure Board(see page 88) and use its Search Area(see page 205) in the JQL mode to run this query:

```
issue in structure("Enterprise Portfolio Testing", "[type = 'Test Case'] and ancestor
in [type = Story and sprint in openSprints()]")
```

Or, you can type only the last part of the query if you use S-JQL search mode(see page 207) on the Structure Board:

```
[type = 'Test Case'] and ancestor in [type = Story and sprint in openSprints()]
```

^ up to the list of examples(see page 429)

3.9.3.5 Find all issues that are blocking critical issues

Setup: Suppose that you have a structure named "Dependency structure" where parent-child relationship corresponds to dependency: each child blocks its parent. (You might have configured a Links Synchronizer(see page 997) to synchronize this structure with the "Dependency" JIRA issue link.)
Let's also suppose that you consider critical those issues that have priority *Critical*.

Goal: You want to see all issues that are blocking critical issues, according to the structure.

How to achieve: You'll need to find children of critical issues. You can use Issue Navigator's Advanced Searching[73] capability or open the structure on the Structure Board(see page 88) and use its Search Area(see page 205) in the JQL mode to run this query:

```
issue in structure("Dependency structure", "child of [priority = Critical]")
```

Or, you can type only the last part of the query if you use S-JQL search mode(see page 207) on the Structure Board:

```
child of [priority = Critical]
```

^ up to the list of examples(see page 429)

72 http://confluence.atlassian.com/display/JIRA/Advanced+Searching
73 http://confluence.atlassian.com/display/JIRA/Advanced+Searching

3.9.3.6 Find all unassigned issues in a part of a project

Setup: Suppose that you use a structure named "Project work breakdown" to break down your project into smaller pieces, so that if you have an issue somewhere in the structure, all of its children at all levels constitute a separate part of a project.

Goal: You are focusing on a part of a project under the issue with key `PROJ-123`, and you want to see unassigned issues in that part of the project.

How to achieve: Use this JQL query to find all unassigned descendants of `PROJ-123`:

```
issue in structure("Project work breakdown", "[assignee is empty] and descendant of
PROJ-123")
```

^ up to the list of examples(see page 429)

3.9.3.7 Top-level view on unfinished parts of a project

Setup: Let's continue with the "Project work breakdown" structure from the previous example. Suppose that there are several top-level issues representing different parts of the project.

Goal: You want to have a view on the parts of the project that are yet unfinished.

How to achieve: In the Structure terms, you need to see the root issues that have unresolved descendants. To have a persistent view, create a Saved Filter[74] with the following JQL:

```
issue in structure("Project work breakdown", "root and descendants in [resolution is
empty]")
```

^ up to the list of examples(see page 429)

3.9.3.8 Find violations of the rule "Tasks must be under Epics or Stories"

Setup: You have a structure named "Planning" where you put issues of types Epic, Story, and Task. Your team follows the convention that Tasks are always put under Epics or Stories. However, as humans are fallible, sometimes a Task ends up being in a wrong place — either on the top level, or under another Task.

Goal: You need to find Tasks that violate the rule, so that you can put them in the right place.

How to achieve: In the Search Area(see page 205) on the Structure Board(see page 88), run the following JQL search(see page 207):

```
issue in structure("Planning", "[type = Task] and parent not in [type in (Epic,
Story)]")
```

^ up to the list of examples(see page 429)

74 https://confluence.atlassian.com/display/JIRA/Using+Filters

3.9.3.9 Find violations of the rule "An issue cannot be resolved if it has unresolved children"

Setup: Suppose that "Planning" is a work breakdown structure. Your team follows the convention that an issue cannot be resolved unless all of its children are resolved.

Goal: You need to find the issues violating this rule.

How to achieve: In the Search Area(see page 205) on the Structure Board(see page 88), run the following S-JQL search(see page 207):

```
[resolution is not empty] and child in [resolution is empty]
```

^ up to the list of examples(see page 429)

3.9.3.10 Find issues that can be resolved because all their children are resolved

Setup: Suppose that "Planning" is a work breakdown structure. Your team follows the convention that once all children of an issue are resolved, the issue can be resolved as well.
The best solution for this would be to use a Status Rollup Synchronizer(see page 1003), but suppose that for some reason you want to do it manually.

Goal: You need a way to manually resolve those issues that have all of their children resolved.

How to achieve: Open the structure on the Structure Board(see page 88). When you paste the query given below into the Search Area(see page 205) (ensure that the JQL mode(see page 207) is selected), the issues that you can resolve will be shown. You can resolve them one by one. Here's the query you need:

```
issue in structure("Planning", "[resolution is empty] and not(child is empty or child
in [resolution is empty])")
```

^ up to the list of examples(see page 429)

3.9.3.11 Get a view of a second (third, ...) level of the hierarchy

Setup: There is a large structure named "Joint Effort" where different users track their issues on several levels: Customer Relations department works with the top-level issues, Project Managers break them down in several issues on the second level, Team Members work with issues under second-level issues.

Goal: Each user wants to see only the relevant part of the structure. Customer Relations department wants to filter out lower-level issues to focus on the top-level ones, and Project Managers sometimes want to focus on just the second-level issues in the context of their parent requests.

How to achieve: use the Search Area(see page 205) on the Structure Board(see page 88) to run the specific queries (ensure that the S-JQL mode(see page 207) is selected.) Toggle the Filter(see page 0) button to hide the issues on the lower levels.

To see top-level issues, run this query:

```
root
```

To see second-level issues (top-level issues will be still displayed, but greyed out), run this query:

```
child of root
```

If you would need to dig even deeper, to see the third level but not the lower ones, you'd use this query:

```
child of (child of root)
```

^ up to the list of examples(see page 429)

3.9.3.12 Get the contents of a folder

Setup: There is a structure with a folder named "Next Release". Issues are placed there manually and then queried via S-JQL for planning purposes (as an Agile board filter, for example).

Goal: The users want to see all issues that are located under the specified folder.

How to achieve: In the Issue Navigator, switch to Advanced Searching[75] and run the following query:

```
issue in structure("My Structure", "descendant of folder('next release')")
```

Note that the folder name is case-insensitive.

^ up to the list of examples(see page 429)

3.9.4 S-JQL Reference

Structure query is a hierarchical condition on the items added to the structure. Structure query is expressed in the Structured JQL language (S-JQL), described in this article.

> Parts of this article assume that you are familiar with Advanced Searching[76] capability of JIRA.

List of Structured JQL topics:

3.9.4.1 Multiple instances of items

If there are multiple instances of an item in the structure, some of these instances might match the query, and some might not.

Consider the following structure:

75 http://confluence.atlassian.com/display/JIRA/Advanced+Searching
76 https://confluence.atlassian.com/display/JIRA/Advanced+Searching

```
TS-239
  TS-42
TS-123
  TS-239
```

Here, issue TS-239 is present two times — one at the root position, and another under another issue. Query `root` wil l match the first instance but not the second one.

This difference is visible when you are filtering in the Structure Widget (see Filter(see page 207)). However, structure() JQL function(see page 455) matches an issue if *at least one* of its instances in the structure matches the S-JQL query. In this example, `issue in structure(root)` will return `TS-239, TS-123.`

3.9.4.2 Constraints

Structure query consists of *constraints*. A constraint matches items in the structure. In the simplest case, the whole structure query consists of a single constraint; for now, we will consider only this case.

There are two types of constraints: *basic* and *relational* constraints.

^ up to the list of S-JQL topics(see page 434)

3.9.4.3 Basic constraint

A basic constraint matches items that satisfy a condition — regardless of their relative positions to other items.

JQL constraint

JQL constraint matches all issues in the structure that satisfy a JQL query. To specify it, specify the JQL query enclosed in square brackets:

```
[status = Open]
```

`leaf` and `root`

This basic constraint matches items that are located at special positions within the structure.

```
leaf
```

```
root
```

The first constraint matches items at the bottom level of the hierarchy, i.e., items that do not have children (sub-items).
The second constraint matches items at the top level of the hierarchy, i.e., items that do not have a parent.

Specific issue

This kind of basic constraint matches just the referenced issues. If some of the issues are not contained within the structure, they are ignored. If none of the issues are contained within the structure, the constraint matches no issues.

You can specify a comma-separated list of issue keys:

```
TS-129, TS-239
```

One issue key:

```
TS-129
```

Issue ID (or a list of them):

```
19320
```

Function constraint (folder, item)

Functions in S-JQL play the same role as in JQL: it is an extension point, so any vendor can develop their own functions to match items in a custom way.

Structure comes bundled with a few functions: *folder* (matching all folders or folders by name) and *item* (matching all items of the specified type or items by name).

Syntax

A function constraint has a *name* and zero or more *arguments*, depending on the function you are using:

```
folder(Urgent)
```

In the example above, function name is *folder* and its argument is *Urgent.*

You can insert any amount of spaces around the name and arguments:

```
folder ( Urgent )
```

Multiple function arguments should be separated by commas:

```
item(Status, In Progress)
```

If an argument contains commas or parentheses, you need to enclose it in "double quotes" or 'single quotes':

```
item(Status, "Done, Sealed, and Delivered")
folder("NU (non-urgent) issues")
```

The former example matches Status items in structure that are named *Done, Sealed, and Delivered.* If this name wasn't enclosed in quotes, the query would mean that function *item* is given four arguments: *Status*, *Done*, *Sealed* and *and Delivered*.
The latter example matches folders named *NU (non-urgent) issues.* If quotes were not used, the query would be incorrect because the first closing parenthesis would be understood as the end of *folder*'s arguments.

If your argument contains quotes, you need to use another type of quotes to enclose it. Suppose that you need to match a version named *3.0, 3.0.1 "Armageddon"*:

```
item(version, '3.0, 3.0.1 "Armageddon"')
```

You can also escape the quotes using backslash (\). Suppose that the version is named *3.0 Beta 1 "Armageddon's Near":*

```
item(version, '3.0 Beta 1 "Armageddon\'s Near"')
```

If you need to use backslash character on its own, you can escape it with another backslash (\\). Suppose that you need to match a folder named *\ (backslash) and related characters*:

```
folder ('\\ (backslash) and related characters')
```

Note that if you don't need to enclose your argument in quotes, then you don't need to escape quotes or backslashes contained within it:

```
folder (Joe's)
folder ( \ )
```

Finally, if there's only one argument and the argument doesn't contain spaces (or is enclosed in quotes), you can omit the parentheses:

```
folder Urgent
folder "Not urgent"
```

folder()

This function matches folder items in the structure, optionally filtering them by name.

Without arguments, this function matches all folders:

```
folder()
```

With one argument, this function matches folders by name (that you see in the *Summary* column). A folder is matched if its name *starts with* the text specified in the first argument. Difference between capital and small letters is ignored.

For example, the following queries match folders named `My issues`, `Issues for Carol`, and `Non-issues`; and do not match folders named `Is suing` or `Issuance`:

```
folder issue
folder Issue
```

If you specify several words separated by spaces, `folder` will match only folders containing all of these words.

> If you're familiar with how Text Search in structure(see page 205) works, then it's useful to think of this argument in the same way as of the simple query. The only difference is that `folder` doesn't recognize issue keys.

There's an advanced matching option for those who like to use *regular expressions.*

To tell `folder` that you are specifying a regular expression, enclose it in slashes (`/`):

```
folder /i.*ue/
```

If the argument starts with a slash but doesn't end with a slash, regular expression matching doesn't occur, and it's matched as a simple text. If you need to write a simple text search where a text starts and ends with a slash, escape the leading slash with a backslash (`\`):

```
folder \/???/
```

The query in the example above matches folder `/???/`.

Another advanced topic is how to query for the exact word (e.g., match `issue` but not `issues`).

This is called *strict searching*. Strict searching is turned on when the *search text* starts and ends with a double quote ("). Note, however, that quotes are stripped off from function arguments, since quoting is also used to allow specifying spaces or parentheses in the search text. Thus you'll need to enclose the search text in single quotes (').

```
folder '"issue"'
```

item()

This function matches items of the specified type in the structure, optionally filtering them by name. It is a generalization of `folder()` function to other item types.

The function takes two arguments: *item type* and *name* (optional). The second argument works in the same way as the argument for `folder()` function.

You can reference either standard item types (provided by Structure plugin) or item types provided by third-party plugins.

If you need to match items of all types, use asterisk (*). The following query finds all items that have the word "Infrastructure" in their Summary, regardless of their type:

```
item(*, Infrastructure)
```

Structure provides the following item types:

```
issue
project
version
version-name
project-component
issuetype
status
resolution
priority
label
user
group
date
cf-option
folder
memo
generator
loop-marker
sprint
missing
tempo-account (when Tempo Timesheets plugin is available)
sd-request-type (when Jira Service Management / Jira Service Desk plugin is
available)
```

Structure.Pages[77] plugin provides the following item types:

```
page
```

Item types provided by third-party plugins are specified similarly. Here's how `item()` function looks up item types:

1. It tries to interpret *type name* argument as referring to an item type provided by Structure and looks it up in the list above.
2. If not found, it looks at all item types provided by all plugins (including Structure itself) and checks if the type name *ends with* the specified text *as a word*. "As a word" means that `page` will match Confluence page item type, but `age` won't. More specifically, the considered word boundaries are hyphen (-), underscore (_) and colon (:).
3. It is an error to specify item type ambiguously, i.e. if there are two item types matching the description. The following forms of *item type* argument allow to specify item type more precisely.
    - Fully qualified item type name, e.g. `com.almworks.jira.structure:type-issue` or `com.almworks.structure.pages:type-confluence-page`.

77 https://wiki.almworks.com/display/pages/Structure.Pages

More generally, the form is `<plugin key>:<type name>`.
- Shortened form of the fully qualified item type name, e.g., `structure:issue` or `pages:page`.
  More generally, the form is `<plugin key part>:<type name part>`.
  When `item()` function looks up item type for the argument, and the argument contains colon (:), the function first tries to interpret is as a fully quailified name. Only if nothing is found, it tries to interpret it as a shortened form.

> Don't confuse "*matching items of some type*" and "*matching issues that have field value equal to that item*". For example, `item(status, Open)` matches *status Open*, not *issues with status Open*. If you need the latter, use JQL constraint: `[status = Open]`.

Empty constraint

An empty constraint matching no items:

```
empty
```

This constraint plays the same role as JQL's `EMPTY` keyword. It is intended to be used as a sub-constraint(see page 444) in relational constraints, which are discussed further.

^ up to the list of S-JQL topics(see page 434)

3.9.4.4 Negation

Any constraint, basic or relational, can be negated using keyword `NOT`. This produces a constraint that matches all items that the original constraint doesn't:

```
not root
```

matches all items that are not top-level items in the structure.

You can always enclose a constraint in parentheses to ease understanding. So, all items in the structure except issues `TS-129` and `TS-239` are matched by this structure query:

```
not (TS-129, TS-239)
```

^ up to the list of S-JQL topics(see page 434)

3.9.4.5 Relational constraint

A basic constraint matches items that satisfy a condition. A relational constraint matches items *related to* items that satisfy a condition. *Related* corresponds to a relationship between positions of items in the structure, like parent-child.
For example,

```
TS-129
```

is a basic constraint that matches a single issue `TS-129`;

```
child in TS-129
```

is a relational constraint matching items that have `TS-129` as a child (sub-item).

Relational constraint has the form `relation`(see page 441) `operator`(see page 442) `subConstraint`(see page 444). Here, `subConstraint` is a constraint on the relatives of items to be matched; other parts of relational constraint are discussed in the following sections.

> Note that the form of relational constraint is similar to the form of JQL clause, `field operator value`. Indeed, let's describe in English a JQL query `type in (Epic, Story)`: it matches issues having *type* that is *in* values *Epic, Story*.
> Now, let's describe in English a structure query `parent in [type = Epic]`: it matches items having *parent* that is *in* constraint "`type = Epic`".
> As you can see, the form that can be used to describe the structure query is similar to that of JQL.

^ up to the list of S-JQL topics(see page 434)

Relations

S-JQL has the following relations:

- `child`: item is a child (sub-item) of another item in the structure.
- `parent`: item is a parent of another item in the structure.
- `descendant`: item is a descendant (sub- or sub-sub-...-item) of another item in the structure.
- `ancestor`: item is an ancestor (parent, parent-of-parent, or parent-of-parent-...-of-parent) of another item in the structure.
- `sibling`: item is a sibling of another item in the structure. Two items are considered siblings if they are under the same parent item.
- `prevSibling`: item is a previous (preceding) sibling of another item in the structure.
  item *A* is a preceding sibling of item *B* if it is a sibling of *B* and *A* is higher than *B* (*A* comes before *B*.)
- `nextSibling`: item is a next (following) sibling of another item in the structure.
  item *A* is a following sibling of item *B* if it is a sibling of *B* and *A* is lower than *B* (*A* comes after *B*.)
- `self` and `issue` are relations of an item (or an issue) to itself. Their role is explained later, in the `self` and `issue` relation(see page 445) section, because at first one has to learn how operators and sub-constraints work.
  There are also combinations of `issue` and `self` with all other relations, listed for completeness below:

`childOrSelf`	`childOrIssue`
`parentOrSelf`	`parentOrIssue`
`descendantOrSelf`	`descendantOrIssue`
`ancestorOrSelf`	`ancestorOrIssue`

`siblingOrSelf`	`siblingOrIssue`
`prevSiblingOrSelf`	`prevSiblingOrIssue`
`nextSiblingOrSelf`	`nextSiblingOrIssue`

Those familiar with XPath may have recognized these relations; indeed, they work like the corresponding XPath axes.

^ up to the list of S-JQL topics(see page 434)

Operators

These are the operators used in S-JQL:

```
IN, NOT IN, IS, IS NOT, =, !=, OF
```

`operator` specifies how `subConstraint` is applied to `relation`:

1. `IN`, `IS`, and `=` put constraint on the relatives of a matched item.

   For example, consider

   ```
   child in (TS-129, TS-239)
   ```

   Here, `relation` is `child`, so an item's relative in question is its child in the structure. Thus, an item matches if *at least one of its children is `TS-129` or `TS-239`*.

   There is no difference between these three operators, unlike JQL. Different forms exist to allow for more natural-looking queries with some sub-constraints.

2. `NOT IN`, `IS NOT`, and `!=` are negated versions of `IN`, `IS`, and `=`. That is, an item is matched if it *is not related to* any item matching `subConstraint`.

   As an important consequence, item that has no relatives is matched.

   For example, consider

   ```
   child not in (TS-129, TS-239)
   ```

An item matches if *no child is TS-129 nor TS-239*. So, this constraint matches all items that either have no children or do not have any of these two items among their children.

Using one of these operators in a relational constraint is the same as using `IN` (or `IS`, or `=`) and negating the whole relational constraint. Thus, the constraint above is equivalent to

```
not (child in (TS-129, TS-239))
```

But, using one of these operators is **very not** the same as using operator `IN` and negating `subConstraint`!

First, *having relatives other than X* is not the same as *not having relatives X*. Think of it as of relationships in a human family: having a relative other than brother (e.g., a sister) is **not** the same as not having a brother, because one may have both a sister and a brother.
Second, an item with no relatives is not matched by the transformed query.

For example,

```
child in (not (TS-129, TS-239))
```

matches all items that have at least one child that is neither `TS-129` nor `TS-239`. That is, the only items that are not matched are leaves and those that have only `TS-129` or `TS-239` as children.

3. `OF` matches the relatives of items that satisfy `subConstraint`.

   For example, consider

   ```
   child of (TS-129, TS-239)
   ```

   An item matches if *it is a child of either TS-129 or TS-239*.

To have a model of how operators `IN` (`IS`, `=`) and `OF` work and to understand the difference between them, consider the table below. Suppose that we take all items in the structure and put each of them, one by one, in column **item**. For each item, we take all of its relatives and put each of them, one by one, in column **relative**. Thus we get pairs of items. We examine each pair, and if one of the components satisfies *subConstraint*, we add the other component to the result set. Which component is added, depends on the operator:

operator	item	relative
`in`	*add to result set*	*satisfies* `subConstraint`
`of`	*satisfies* `subConstraint`	*add to result set*

One may note that for any relation, there is a corresponding "inverse": for example, `child` is the inverse of `parent`, and vice versa. A relational constraint that uses operator `IN` (`IS`, `=`) is equivalent to a relational constraint that uses an inverse relation with operator `OF`. That is,

```
child in (TS-129, TS-239)
```

is the same as

```
parent of (TS-129, TS-239)
```

Again, different forms of expressing the same constraint exist to allow for more natural-looking queries.

^ up to the list of S-JQL topics(see page 434)

Sub-constraints

Any constraint can be used as a sub-constraint, whether basic, relational, or a combination of those(see page 446). For example,

```
child of root
```

selects items on the second level of the hierarchy. To select items on the third level of the hierarchy, you can once again use relation `child` and the previous query as `subConstraint`:

```
child of (child of root)
```

There is a special basic constraint, `empty`, which matches no items. It is used as a sub-constraint to match items that have no relatives as per `relation`.

For example, let's take relation `child` and see what the corresponding relational constraints with different operators mean.

`child is empty`	matches all items that have no children (equivalent of `leaf`)
`child is not empty`	matches all items that have at least one child (equivalent of `not leaf`)
`child of empty`	matches all items that are not children of other items (equivalent of `root`)

Of course, using `leaf` or `root` is more convenient, but you can apply `empty` to any other relation. For instance, `sibling is empty` matches an item if it is the only child of its parent.

^ up to the list of S-JQL topics(see page 434)

`self` and `issues` relations: adding sub-constraint matches to the result set

A relational constraint with relation `self` behaves exactly as its sub-constraint, possibly negated if operator `NOT IN` (`IS NOT`, `!=`) is used.
Thus,

```
self in [status = Open]
```

is equivalent to

```
[status = Open]
```

Similarly,

```
self not in [status = Open]
```

is equivalent to

```
not [status = Open]
```

When combined with another relation, `self` allows to add the items matched by `subConstraint` to the resulting set. For example,

```
descendant of TS-129
```

returns all of the children of `TS-129` at all levels, but does not return `TS-129` itself. To add `TS-129`, use `descendantOrSelf`:

```
descendantOrSelf of TS-129
```

`issue` relation

`issue` is a special case of `self` relation that only matches issues. For instance, if on the top level of the structure you have folders and issues, and you want to hide all folders, you can write this:

```
descendantOrIssue of root
```

This query matches all top-level issues and all their sub-items.

^ up to the list of S-JQL topics(see page 434)

3.9.4.6 Combining constraints with Boolean operators

We can now define a structure query as a *Boolean combination of constraints*, that is, a structure query consists of constraints connected with `AND` and `OR`. When two constraints are connected with `AND`, together they will match issues that are matched by both constraints. This allows you to limit the results. Likewise, when two constraints are connected by `OR`, together they will match issues that are matched by at least one of the constraints. This allows you to expand the results.

Note that `AND` has higher precedence than `OR`. That means that the Structure query

```
leaf or (parent of leaf) and [status = Open]
```

matches all issues that are either leaves, or are parents of leaves in status *Open*. In order to also constrain leaf issues to be in the status *Open*, you need to use parentheses:

```
(leaf or (parent of leaf)) and [status = Open]
```

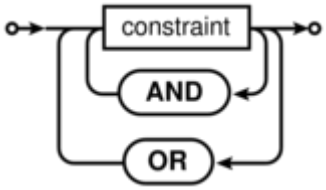
^ up to the list of S-JQL topics(see page 434)

3.9.4.7 Railroad diagrams

As a final piece of reference, here's the S-JQL syntax in the form of railroad diagrams[78].

> ⓘ S-JQL keywords are not case-sensitive, and all underscores in keywords are optional.

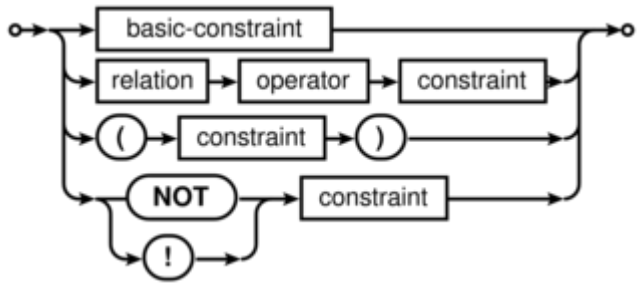
structure-query



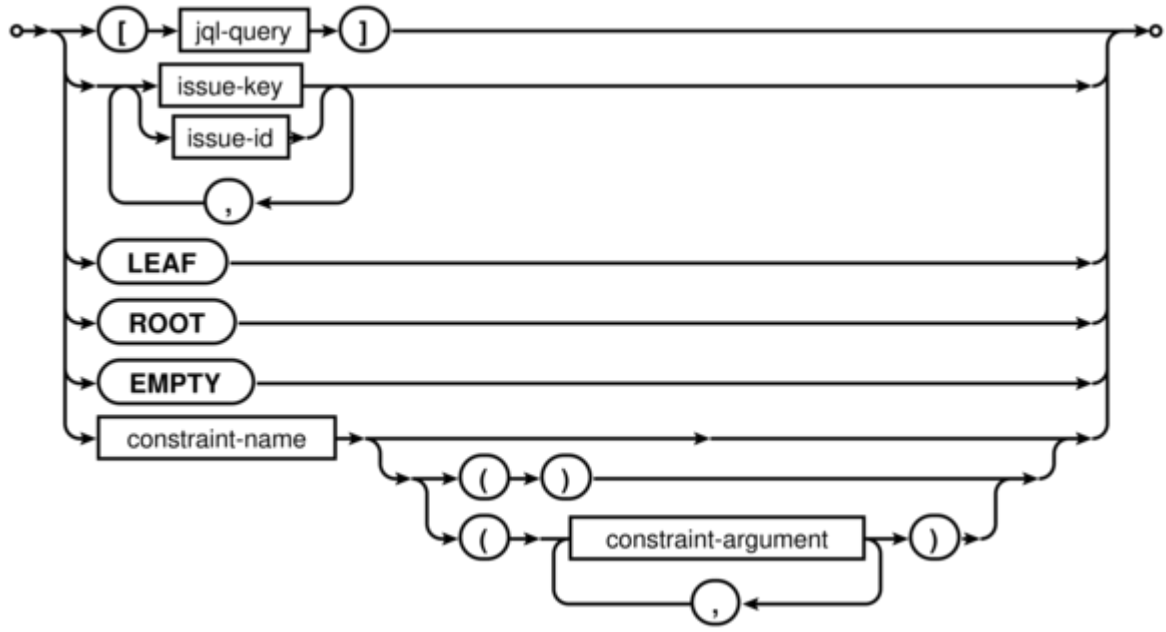
> ⓘ S-JQL admits using && and & in place of `AND`, as well as || and | in place of `OR`.

78 http://en.wikipedia.org/wiki/Railroad_diagram

constraint



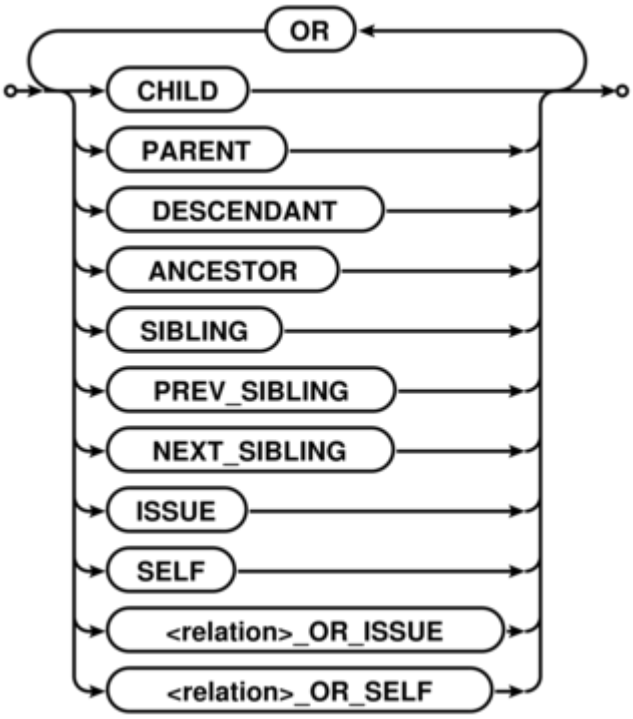
basic-constraint



- `jql-query` is any valid JQL query.
- `issue-key` is any valid JIRA issue key.
- `issue-id` is any valid JIRA issue ID.
- `constraint-name` is the name of the function constraint: either bundled with Structure (`folder`, `item`, or `row_id`) or provided by a Structure extension (plugin).
- `constraint-argument` is one of the following:
  - either a sequence of non-whitespace characters
  - or quoted text (inside "double quotes" or 'single quotes'), where quotes can be escaped via backslash: `\"`, `\'`; backslash itself can be escaped: `\\`.

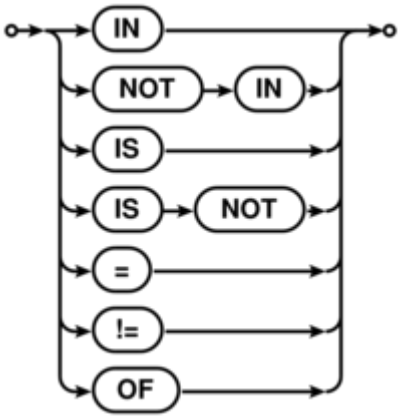
  See also Function constraint - Syntax(see page 0).

relation



S-JQL admits using || and | in place of OR.

operator



^ up to the list of S-JQL topics(see page 434)

3.9.4.8 List of S-JQL keywords

In this article, all S-JQL keywords are listed in all of their spelling variants. This is intended for developers creating their own S-JQL function, because function name must not coincide with the existing keyword.

```
!
!=
&
&&
(
)
,
=
[
]
ancestor
ancestor_or_issue
ancestor_or_issues
ancestor_or_self
ancestorOrIssue
ancestorOrIssues
ancestorOrSelf
ancestors
ancestors_or_issue
ancestors_or_issues
ancestors_or_self
ancestorsOrIssue
ancestorsOrIssues
ancestorsOrSelf
and
child
child_or_issue
child_or_issues
child_or_self
childOrIssue
childOrIssues
childOrSelf
children
children_or_issue
children_or_issues
children_or_self
childrenOrIssue
childrenOrIssues
childrenOrSelf
descendant
descendant_or_issue
descendant_or_issues
descendant_or_self
descendantOrIssue
descendantOrIssues
descendantOrSelf
descendants
descendants_or_issue
descendants_or_issues
descendants_or_self
descendantsOrIssue
```

```
descendantsOrIssues
descendantsOrSelf
empty
following_sibling
following_sibling_or_issue
following_sibling_or_issues
following_sibling_or_self
following_siblings
following_siblings_or_issue
following_siblings_or_issues
following_siblings_or_self
followingSibling
followingSiblingOrIssue
followingSiblingOrIssues
followingSiblingOrSelf
followingSiblings
followingSiblingsOrIssue
followingSiblingsOrIssues
followingSiblingsOrSelf
in
is
issue
issue_or_ancestor
issue_or_ancestors
issue_or_child
issue_or_children
issue_or_descendant
issue_or_descendants
issue_or_following_sibling
issue_or_following_siblings
issue_or_next_sibling
issue_or_next_siblings
issue_or_parent
issue_or_parents
issue_or_preceding_sibling
issue_or_preceding_siblings
issue_or_prev_sibling
issue_or_prev_siblings
issue_or_previous_sibling
issue_or_previous_siblings
issue_or_sibling
issue_or_siblings
issue_or_sub_issue
issue_or_sub_issues
issueOrAncestor
issueOrAncestors
issueOrChild
issueOrChildren
issueOrDescendant
issueOrDescendants
issueOrFollowingSibling
issueOrFollowingSiblings
```

```
issueOrNextSibling
issueOrNextSiblings
issueOrParent
issueOrParents
issueOrPrecedingSibling
issueOrPrecedingSiblings
issueOrPreviousSibling
issueOrPreviousSiblings
issueOrPrevSibling
issueOrPrevSiblings
issueOrSibling
issueOrSiblings
issueOrSubIssue
issueOrSubIssues
issues
issues_or_ancestor
issues_or_ancestors
issues_or_child
issues_or_children
issues_or_descendant
issues_or_descendants
issues_or_following_sibling
issues_or_following_siblings
issues_or_next_sibling
issues_or_next_siblings
issues_or_parent
issues_or_parents
issues_or_preceding_sibling
issues_or_preceding_siblings
issues_or_prev_sibling
issues_or_prev_siblings
issues_or_previous_sibling
issues_or_previous_siblings
issues_or_sibling
issues_or_siblings
issues_or_sub_issue
issues_or_sub_issues
issuesOrAncestor
issuesOrAncestors
issuesOrChild
issuesOrChildren
issuesOrDescendant
issuesOrDescendants
issuesOrFollowingSibling
issuesOrFollowingSiblings
issuesOrNextSibling
issuesOrNextSiblings
issuesOrParent
issuesOrParents
issuesOrPrecedingSibling
issuesOrPrecedingSiblings
issuesOrPreviousSibling
```

```
issuesOrPreviousSiblings
issuesOrPrevSibling
issuesOrPrevSiblings
issuesOrSibling
issuesOrSiblings
issuesOrSubIssue
issuesOrSubIssues
leaf
leaves
next_sibling
next_sibling_or_issue
next_sibling_or_issues
next_sibling_or_self
next_siblings
next_siblings_or_issue
next_siblings_or_issues
next_siblings_or_self
nextSibling
nextSiblingOrIssue
nextSiblingOrIssues
nextSiblingOrSelf
nextSiblings
nextSiblingsOrIssue
nextSiblingsOrIssues
nextSiblingsOrSelf
not
null
of
or
parent
parent_or_issue
parent_or_issues
parent_or_self
parentOrIssue
parentOrIssues
parentOrSelf
parents
parents_or_issue
parents_or_issues
parents_or_self
parentsOrIssue
parentsOrIssues
parentsOrSelf
preceding_sibling
preceding_sibling_or_issue
preceding_sibling_or_issues
preceding_sibling_or_self
preceding_siblings
preceding_siblings_or_issue
preceding_siblings_or_issues
preceding_siblings_or_self
precedingSibling
```

```
precedingSiblingOrIssue
precedingSiblingOrIssues
precedingSiblingOrSelf
precedingSiblings
precedingSiblingsOrIssue
precedingSiblingsOrIssues
precedingSiblingsOrSelf
prev_sibling
prev_sibling_or_issue
prev_sibling_or_issues
prev_sibling_or_self
prev_siblings
prev_siblings_or_issue
prev_siblings_or_issues
prev_siblings_or_self
previous_sibling
previous_sibling_or_issue
previous_sibling_or_issues
previous_sibling_or_self
previous_siblings
previous_siblings_or_issue
previous_siblings_or_issues
previous_siblings_or_self
previousSibling
previousSiblingOrIssue
previousSiblingOrIssues
previousSiblingOrSelf
previousSiblings
previousSiblingsOrIssue
previousSiblingsOrIssues
previousSiblingsOrSelf
prevSibling
prevSiblingOrIssue
prevSiblingOrIssues
prevSiblingOrSelf
prevSiblings
prevSiblingsOrIssue
prevSiblingsOrIssues
prevSiblingsOrSelf
root
roots
self
self_or_ancestor
self_or_ancestors
self_or_child
self_or_children
self_or_descendant
self_or_descendants
self_or_following_sibling
self_or_following_siblings
self_or_next_sibling
self_or_next_siblings
```

```
self_or_parent
self_or_parents
self_or_preceding_sibling
self_or_preceding_siblings
self_or_prev_sibling
self_or_prev_siblings
self_or_previous_sibling
self_or_previous_siblings
self_or_sibling
self_or_siblings
self_or_sub_issue
self_or_sub_issues
selfOrAncestor
selfOrAncestors
selfOrChild
selfOrChildren
selfOrDescendant
selfOrDescendants
selfOrFollowingSibling
selfOrFollowingSiblings
selfOrNextSibling
selfOrNextSiblings
selfOrParent
selfOrParents
selfOrPrecedingSibling
selfOrPrecedingSiblings
selfOrPreviousSibling
selfOrPreviousSiblings
selfOrPrevSibling
selfOrPrevSiblings
selfOrSibling
selfOrSiblings
selfOrSubIssue
selfOrSubIssues
sibling
sibling_or_issue
sibling_or_issues
sibling_or_self
siblingOrIssue
siblingOrIssues
siblingOrSelf
siblings
siblings_or_issue
siblings_or_issues
siblings_or_self
siblingsOrIssue
siblingsOrIssues
siblingsOrSelf
sub_issue
sub_issue_or_issue
sub_issue_or_issues
sub_issue_or_self
```

```
sub_issues
sub_issues_or_issue
sub_issues_or_issues
sub_issues_or_self
subIssue
subIssueOrIssue
subIssueOrIssues
subIssueOrSelf
subIssues
subIssuesOrIssue
subIssuesOrIssues
subIssuesOrSelf
|
||
```

3.9.5 structure() JQL function

Structure adds `structure()` JQL function that lets you search for issues that are added to a structure, with the possibility to add constraints on their relationships.You can use this function in any place in JIRA where you can use JQL: in the Issue Navigator, in a Saved Filter, as an Agile Board query etc. For more information, see JIRA documentation on Advanced Searching[79] and Advanced Searching Functions[80].

> If a user does not have access to structure(see page 955), they will not be able to create new queries with the `structure()` function and existing queries will have `structure()` function return an empty set. However, the user will still see `structure()` function offered in the JQL completion drop-down.

To specify a structure condition in JQL, use the following format:

issue in structure(*structureName*$_{opt}$, *structureQuery*$_{opt}$**)**

Function arguments:

structureName	*Optional*	The name of the structure. If you omit the structure name, system-wide Default Structure(see page 537) will be searched.
structureQuery	*Optional*	Use this parameter to select only a part of the structure. This parameter specifies a *Structure Query* in a language similar to JQL, Structured JQL(see page 428).

> You can use structure ID instead of the structure name. You can see structure ID in the URL of the Structure Board if you open **Manage Structure** page and click structure name.

79 https://confluence.atlassian.com/display/JIRA/Advanced+Searching
80 https://confluence.atlassian.com/display/JIRA/Advanced+Searching+Functions

3.9.5.1 Function arguments need to be quoted if they contain spaces or non-letters

As dictated by the syntax of JQL, you'll need to enclose structure name or structure query in 'single quotes' or "double quotes" if they contain spaces or non-letters.

What if *structure name* or *structure query* itself contains quotes?

If structure name or structure query contains quotes of one kind, you need to enclose them with a different kind of quotes. That is, if structure query contains double quote, you'll need to enclose it in single quotes. Alternatively, you can escape quote with a backslash: `\"`.

Example 1

Suppose you need to find all issues that are directly under issues in status *Awaiting Deployment.*

In plain JQL, issues in this status can be found via this query: `Status = "Awaiting Deployment"`. Note that since status name contains spaces, JQL requires us to enclose it in quotes.

According to S-JQL Reference(see page 434), the corresponding Structure query would be `child of [Status = "Awaiting Deployment"]`.

That means that you need to enclose this Structure query with single quotes:

```
issue in structure("My personal structure", 'child of [Status = "Awaiting
Deployment"]')
```

Note that the following will **not** work:

```
issue in structure("My personal structure", "child of [Status = "Awaiting
Deployment"]")
```

Example 2: escaping with backslash

In the following example, the query returns issues that are directly under issues assigned to fix version named *3.0 "Armageddon"*.

```
issue in structure("My personal structure", "child of [fixVersion = '3.0
\"Armageddon\"']")
```

3.9.5.2 Backward compatibility with `structure()` JQL function prior to Structure 2.4

Prior to Structure 2.4, `structure()` JQL function did not take structure query as an argument; you could specify only one issue key or ID, and you would get the referenced issue along with all of its children at all levels. As you might have noticed, this old-style usage can be interpreted as a structure query, but according to the rules of S-JQL, it would return just the referenced issue without its children. To maintain backward compatibility, any structure

query in Structure 2.4 that consists of a single basic constraint that references issues by their keys or IDs matches not only these issues, but all of their children as well.

That means that if you were using JQL of the form

```
issue in structure("My personal structure", TS-129)
```

then in Structure 2.4 this query will still return `TS-129` and all of its children at all levels (provided that `TS-129` is added to the structure.)

If this backward compatibility bites you (if, say, you need to check whether an issue is added to a structure), prepend the structure query with `issue in`:

```
issue in structure("My personal structure", "issue in TS-129")
```

This JQL will match only `TS-129` if it is in the structure.

3.10 Columns and Views

A **view** defines which **columns** are displayed in Structure and in what configuration.

Sorry, the widget is not supported in this export.
But you can reach it using the following URL:

https://www.youtube.com/watch?v=WXTEPs-hPyU

The following sections will show you how to customize your columns and views to provide exactly the information you need.

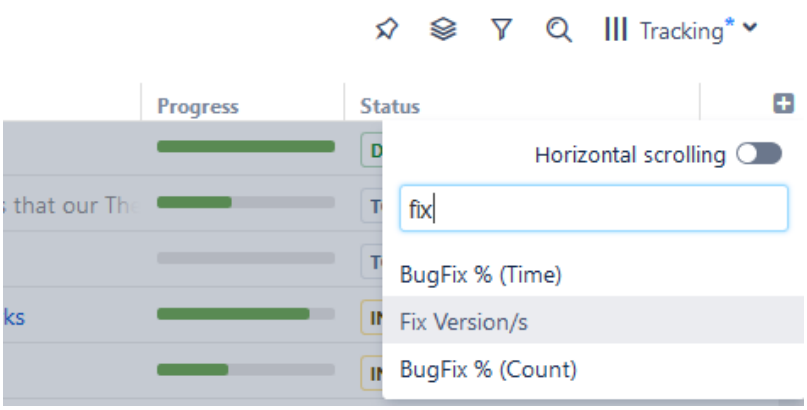
3.10.1 Adding Columns

Structure provides a number of columns that display information about issues in the structure. You can customize(see page 501) the displayed columns by adding new columns, changing each column configuration, or switching to a new view(see page 507).

3.10.1.1 Adding New Columns

To add a column, click on the **+** button at the right corner of the table header. A drop-down with the available column presets appears. To select the desired column, you can:

- Use the mouse to find a specific column,
- Use the keyboard **arrow keys** to select the column and hit **Enter** when done, or
- Start typing the column name to filter the list, and then select the appropriate column.



To abort adding a new column, hit **Escape**.

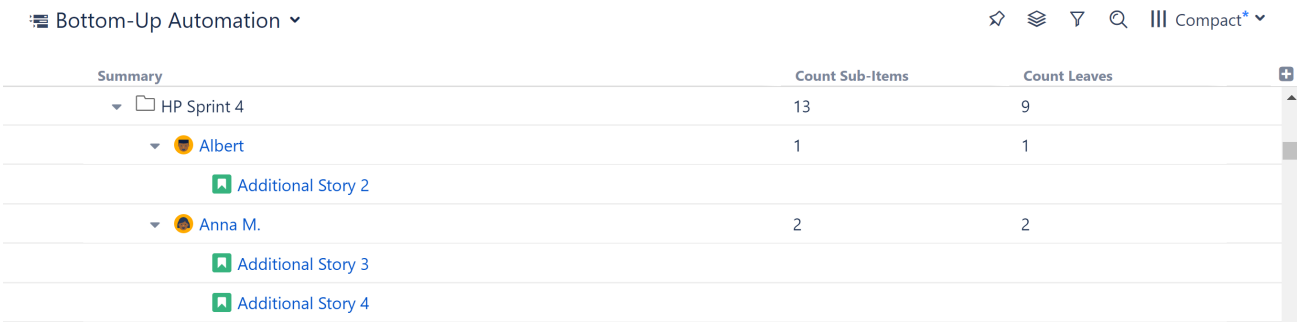
> Use keyboard shortcut **TT** to quickly open the Add Column dialog (hit "t" twice).

To learn more about the different columns available in Structure and how to manage them, see the following articles:mns:

Structure also contains extension API(see page 1029), so the selection of available columns may be extended by a third-party plugin.

3.10.1.2 Count Leaves Column

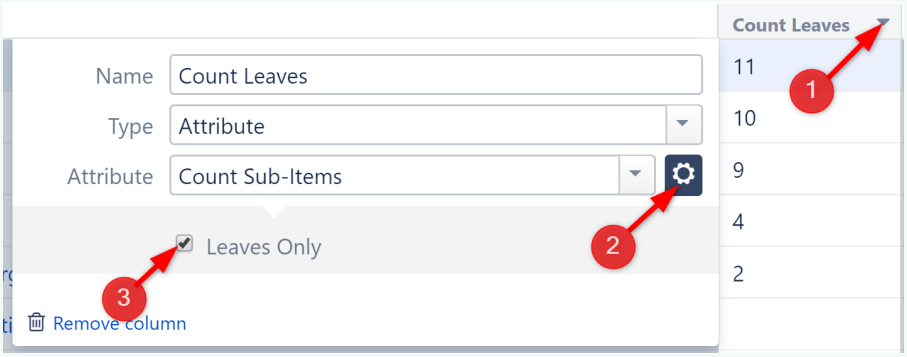
The Count Leaves column allows you to quickly see how many sub-items of an issue are at the last level of the hierarchy (they have no sub-items of their own). This is particularly useful when you have organized issues using Group generators(see page 169) or folders(see page 104) and you need a count of just the issues themselves.



Notice in the example above that HP Sprint 4 has 13 sub-items, but only 9 leaves. The Count Leaves column allows you to omit group items (in this case, the Assignee groups) and look just at the number of issues in the last level of the structure's hierarchy.

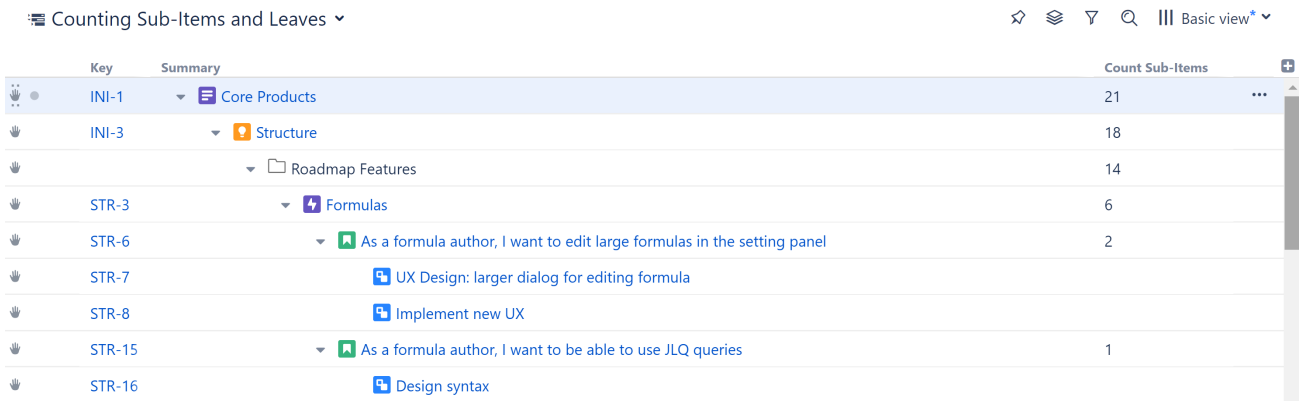
See also: Count Sub-Items Column(see page 460)

You can toggle between Count Sub-Items and Count Leaves. Open the column configuration, click the settings icon and check/uncheck **Leaves Only**.



3.10.1.3 Count Sub-Items Column

The Count Sub-Items column allows you to quickly see how many items are beneath an issue or folder within a structure.



In the example above, we can see that there are 2 sub-items beneath story STR-6 and a total of 14 sub-items contained in the Roadmap Features folder.

Use Cases

The Count Sub-Items column can be used to identify:

- How many items are contained within a folder or other grouping
- How many issues have been assigned to each resource (when your issues are grouped by Assignee)
- How many tasks are necessary to complete each story, epic, or initiative

Depending on how you've grouped your items, you may not want to count all sub-items. Instead, it may be more useful to just count the bottom level. In this case, you could use the Count Leaves column(see page 459).

3.10.1.4 Flags Column

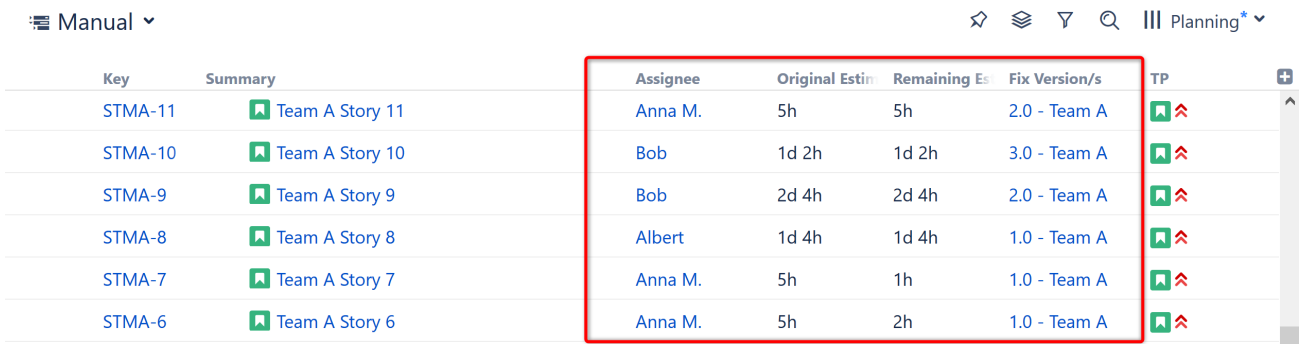
Flags are the small icons displayed at the left side of issue rows to mark specific issue states.

Structure displays the following flags:

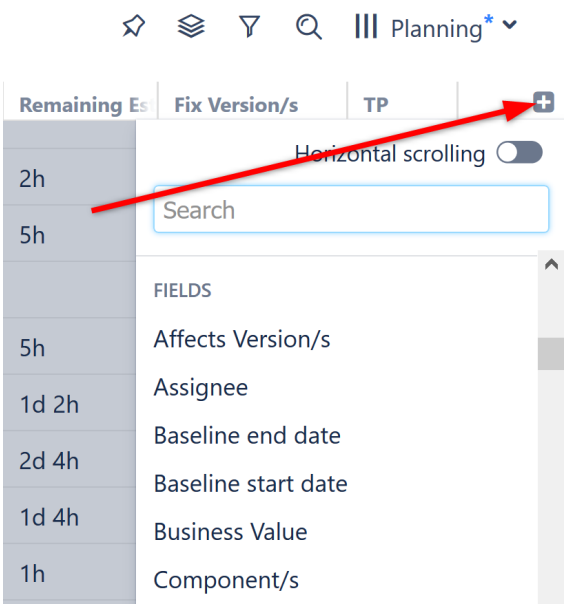
✔	**Resolved** flag means that the issue is in a Done status category. Such issues are considered completed and filtered out by the Unresolved Quick Transformation(see page 220).
🔒	**Read-only** flag means that the current user does not have permission to add, remove or rearrange sub-items of the flagged issue. This flag is only displayed when: • The structure is configured to require Edit Issue permission on Parent Issue(see page 0), • The user does not have permission to edit this issue(see page 132).

3.10.1.5 Field Columns

For each issue field in your Jira, Structure offers a column that displays that field's value.

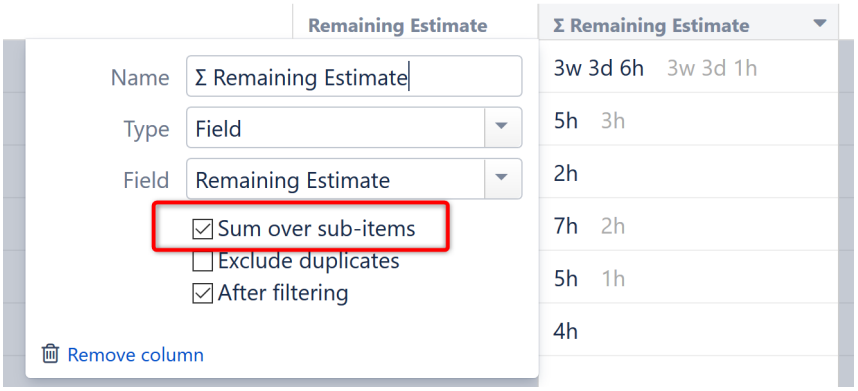


Each Structure view(see page 505) contains a different set of field columns, with a focus on a specific task or business need. But each team has different needs - if you don't see the issue fields you're looking for, you can simply add new columns(see page 501) to include them in your structure.



Displaying Aggregate Values (Totals)

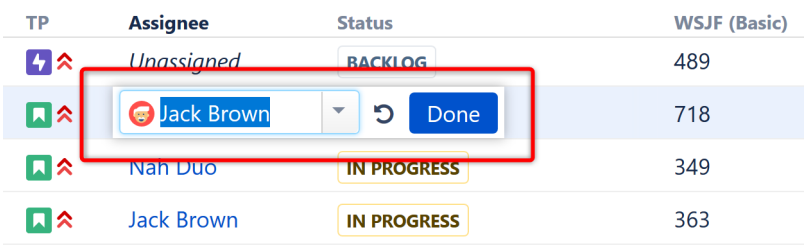
To display aggregate totals(see page 497) for a numeric or time-tracking field, check the **Sum over sub-items** box.



If the Sum over sub-issues option is unavailable for a given field, then the aggregate cannot be calculated.

Editing Field Values

Most issue fields can be edited directly in the Structure panel – you can edit a field's value(see page 132) by double-clicking it (if the field is added to the Edit Screen in Jira).

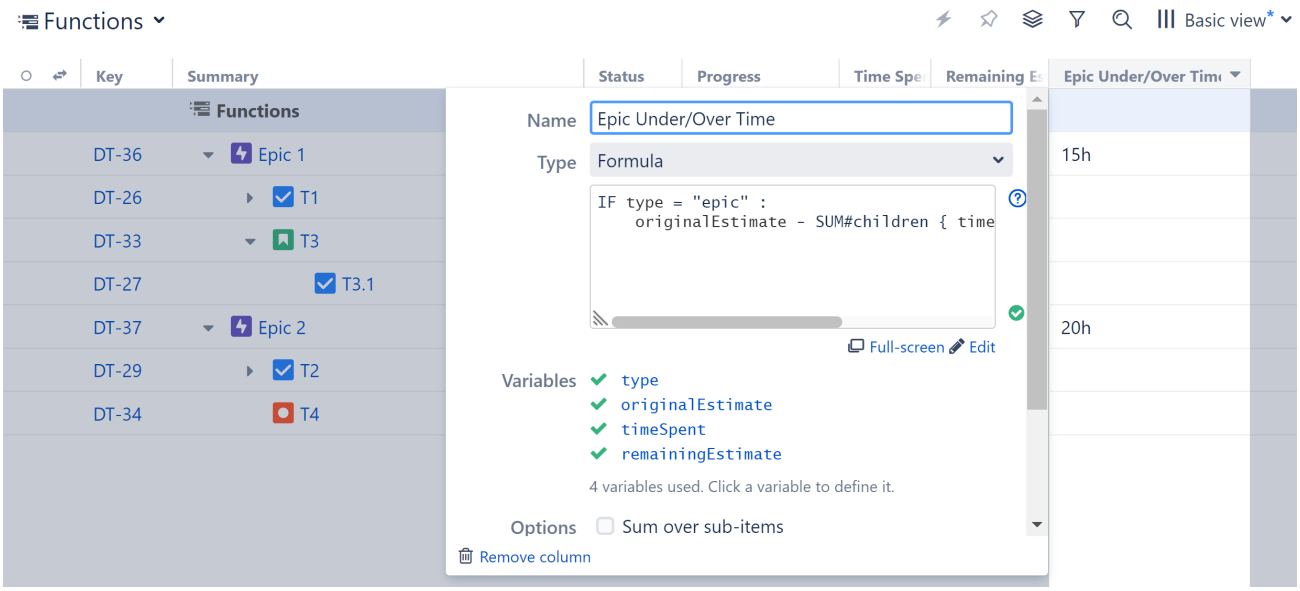


When an aggregate value is displayed, double-clicking the field will allow you to edit the issue's own value.

> ⚠ When editing Status, you can only select statuses that are allowed according to the workflow and that have no required fields or dialogs to show in the corresponding transition.

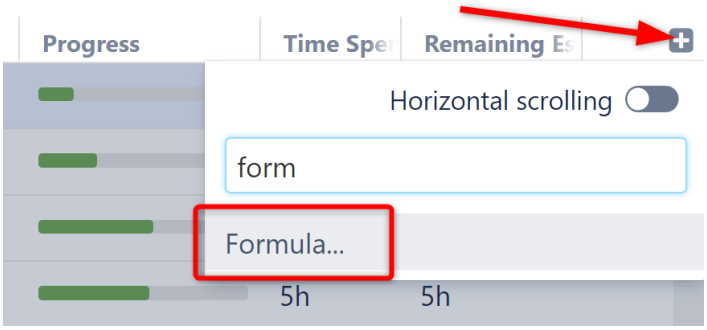
3.10.1.6 Formula Column

Formula columns allow you to perform calculations or comparisons(see page 228) based on issue fields or other attributes, and display those results in a column.



Add a Formula Column

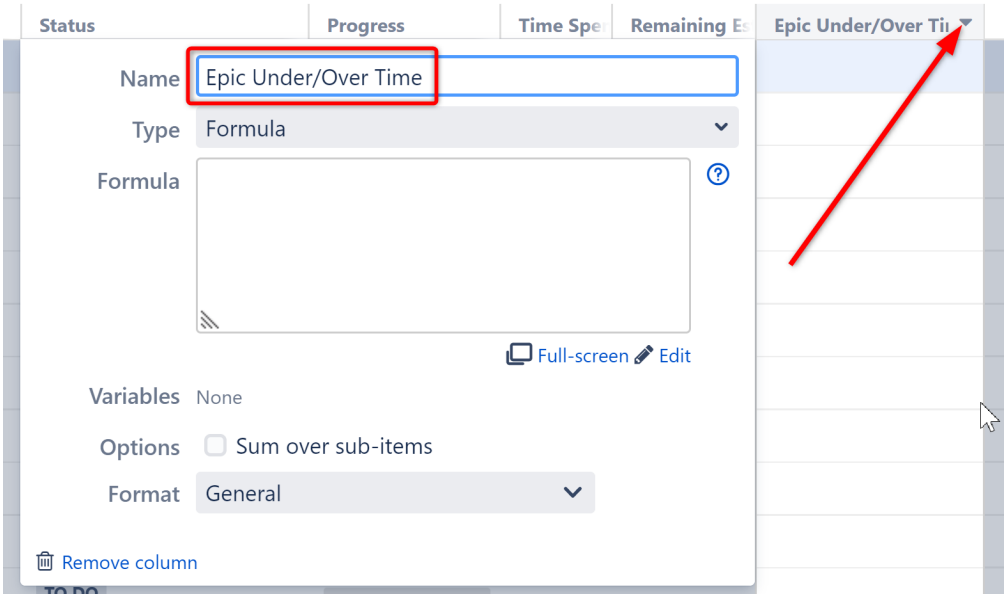
Start with adding a new column(see page 501) by clicking the + icon to the right of the column headings.
Select **Formula...** as its type.



Name Your Formula

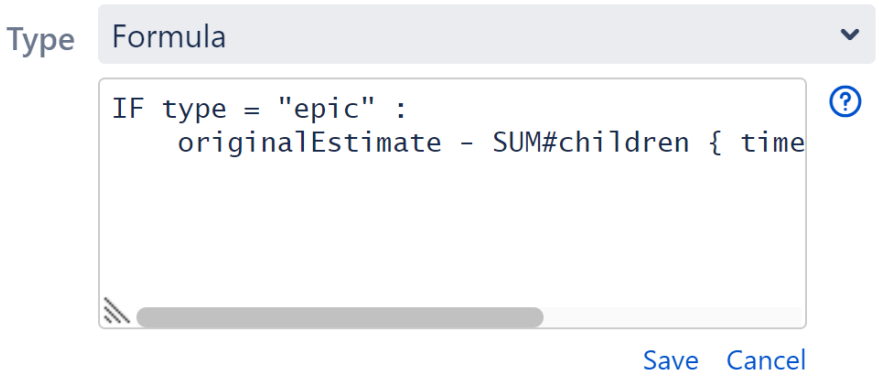
Open the Formula editor by clicking the downward-pointing triangle next to the new column header.

In the **Name** field, give the column a meaningful name – something that expresses the purpose of your formula and will be easily recognized by yourself and anyone you might share it with.



Formula

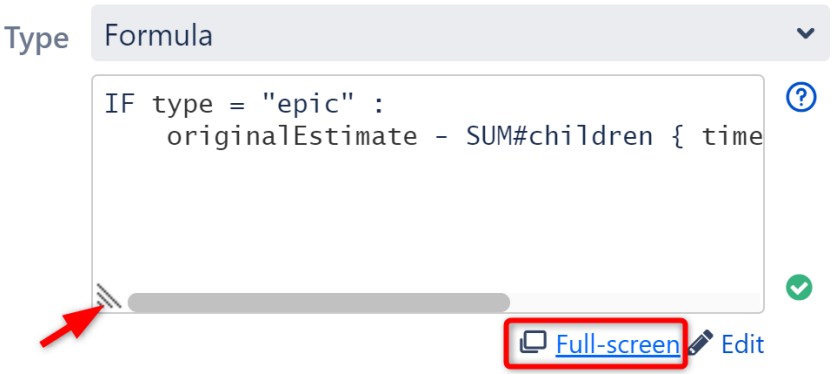
Enter your formula into the **Formula** field and assign any necessary variables.



See Formulas(see page 228) for more information.

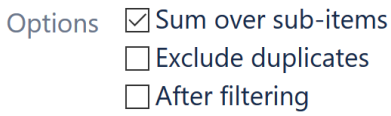
Full Screen

The initial sizing for the formula editor is rather small. If you need more space, drag the corner of the field or click **Full-Screen** to expand the window.



Options (Aggregation)

Select **Sum over sub-items** to have each row display an **aggregate total**, meaning the results for each row will be calculated as a sum of the values for that row and its sub-items.



When aggregation is enabled, you have a couple of options:

- **Exclude duplicates** - If an item appears more than once in the structure, it's value will only be included once within the aggregate total.
- **After filtering** - When checked, filtered items(see page 207) will not be included in the aggregate total. If this is left unchecked, the values of those items will be included in the calculation, even though they are not visible in the structure. *Note: items filtered by a Filter Generator are always excluded.*

You can also use aggregate functions(see page 387) to accomplish the same thing - or to create custom aggregations.

Sum over sub-items doesn't work for all formulas. For example, string values usually cannot be added together.

However, Structure has no way of knowing what each value represents, so these options are always available. When selecting this option, be careful to verify that the calculated values will make sense.

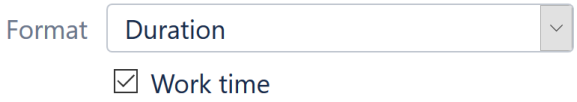
Format

The **Format** section allows you to customize the format of your results. The following options are available:

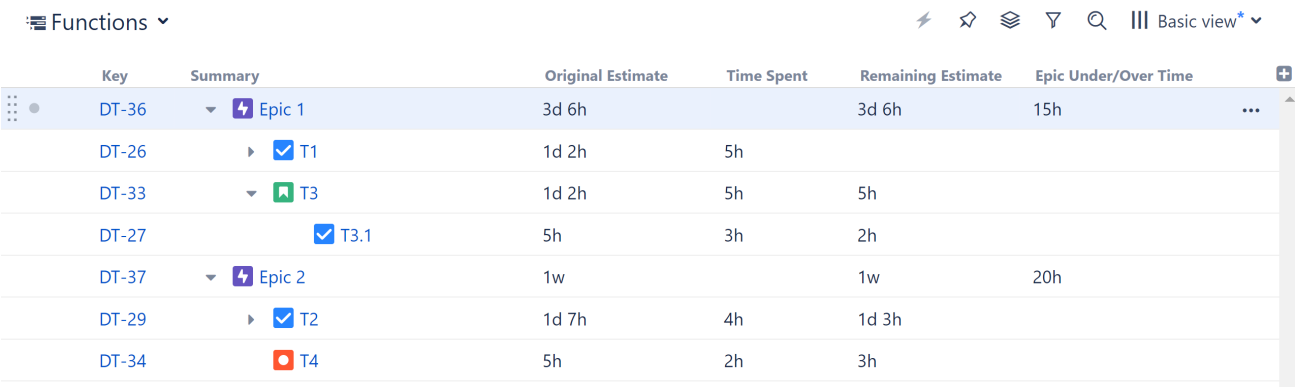
- **General** - this default option will work for most formulas. If your results don't look right, try one of the others.
- **Number -** lets you specify the number of decimal places that will always be shown. The value will be rounded up to the least meaningful digit in this format.
- **Percentage** - treats the value as a ratio (0.0 = 0%, 1.0 = 100%) and adds a percent sign.
- **Date/Time** - displays the results as date/time and allows you to pick the appropriate format.

- **Duration** - displays duration values as days, hours and minutes. You can also select **Work time** to display values using Jira's time tracking settings, so the duration reflects your work hours. See Work Time in Formula Columns(see page 419) for more information.
- **Wiki Markup** - allows you to add wiki markup, including colors and images, to a column. See Wiki Markup in Formula Columns(see page 251) for more details.

Going back to our sample formula, let's change the Format to **Duration** and check **Work time.**



Our new column (Epic Under/Over Time) now displays the weeks, days, and hours that we are either ahead of schedule or behind schedule for each epic:



Key	Summary	Original Estimate	Time Spent	Remaining Estimate	Epic Under/Over Time
DT-36	Epic 1	3d 6h		3d 6h	15h
DT-26	T1	1d 2h	5h		
DT-33	T3	1d 2h	5h	5h	
DT-27	T3.1	5h	3h	2h	
DT-37	Epic 2	1w		1w	20h
DT-29	T2	1d 7h	4h	1d 3h	
DT-34	T4	5h	2h	3h	

Note that dates, times and durations are all numbers in the Expr language.

Unless you select an appropriate format, duration is represented as the number of milliseconds. Dates are represented as "Epoch milliseconds", the number of milliseconds between midnight January 1st, 1970 (GMT) and the specified date, not counting leap seconds. Negative values are allowed to represent earlier dates.

Additional Information

Sharing Formula Columns

Formula columns can be shared by:

- Saving the formula as a global Saved Column(see page 524)
- Making them a part of a public or shared View(see page 511), which other users can select
- Creating a perspective URL(see page 97) that will open the structure with the same configuration, including the formula column

Sorting by Calculated Value

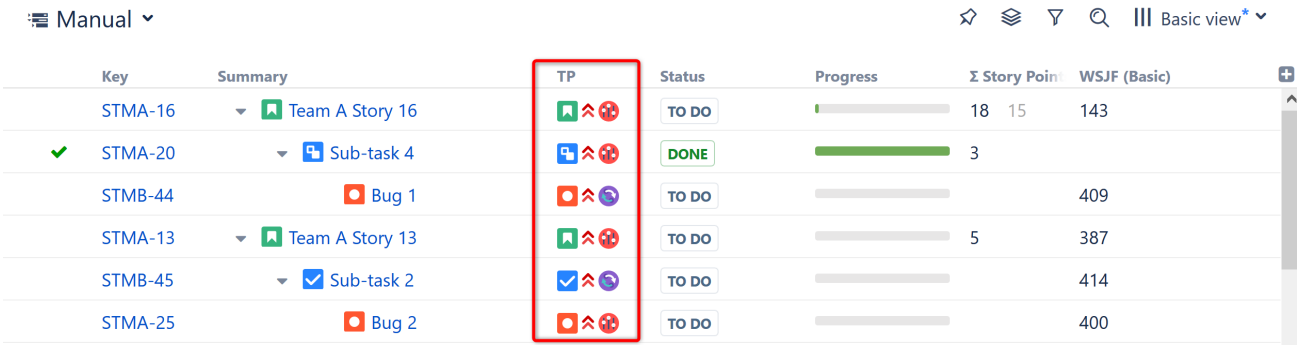
You can sort(see page 74) by the values calculated in a formula column by clicking the column header.

See Also

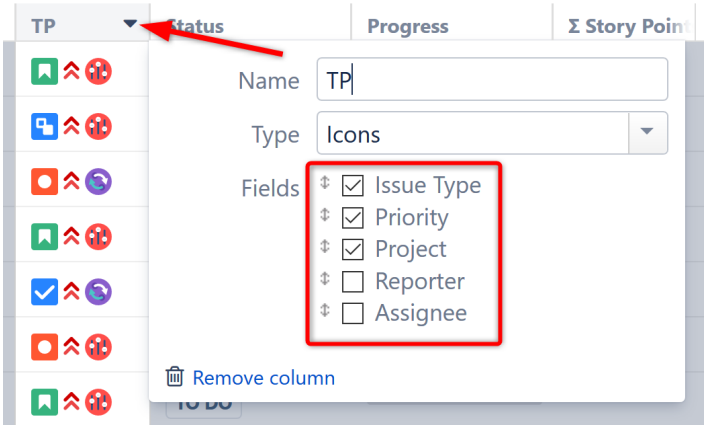
- Formulas(see page 228)
- Bundled Formulas(see page 257)
- Expr Language(see page 235)
- Creating an Advanced Formula Column

3.10.1.7 Icons Column

The Icons column displays icons for issue type, priority, project, reporter and assignee.

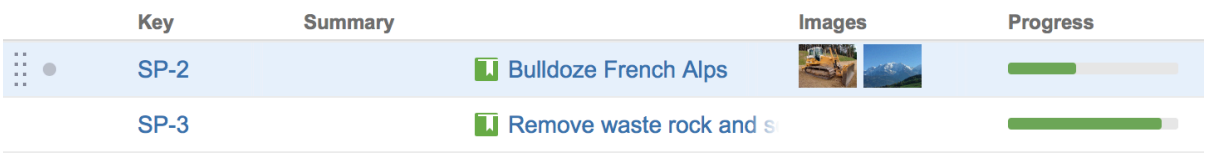


You can choose which icons are displayed and what order they appear in. Simply click the downward-facing arrow next to the column header and select which fields to include. To rearrange your icons, drag and drop the items in the Fields list.



3.10.1.8 Images Column

The Images column displays small thumbnails of attached image files and allows users to view those images in a pop-up dialog.



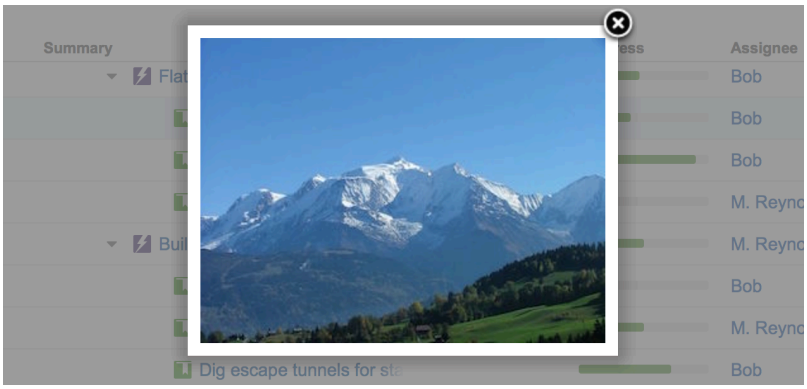
Viewing Full-size Images

Using your mouse:

1. Click the image thumbnail to see the full-size image in a dialog box.
2. If more than one image file is attached, use the left and right arrows to move between them.
3. Click the close button at the top right corner to close the full-size image view.

Using your keyboard:

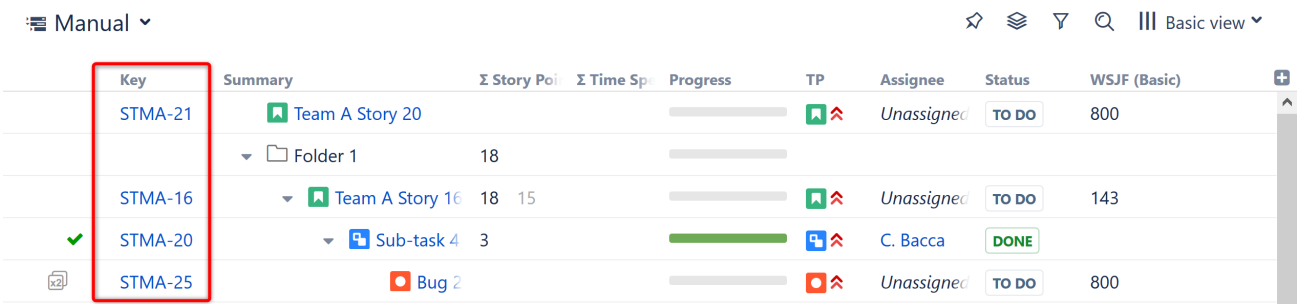
1. Select the issue that contains images.
2. Press **i,i** ("i" twice) to view the first image.
3. Press → to go to the next image, ← to go to the previous image.
4. Press **Esc** to close the full-size image view.



Images from Wikipedia

3.10.1.9 Issue Key Column

The Issue Key column displays the issue key for issues. For other types of items it remains empty.



Compact View

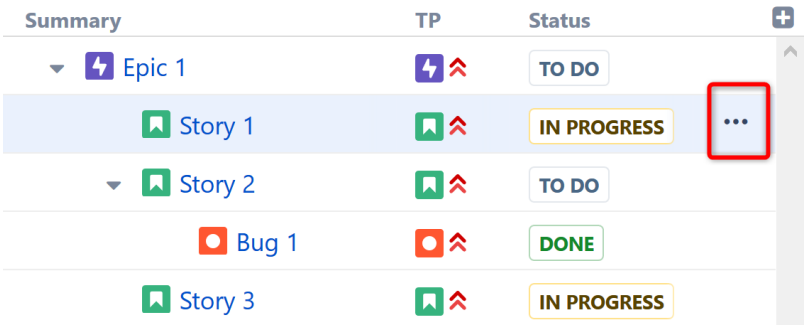
If the project key is large, the issue key column may get too wide. You can configure the Issue Key column to replace the project key with a small avatar icon of the project.

To enable compact view for the Issue Key column, open the column options[81] and select **Compact View.**

81 https://wiki.almworks.com/display/structure/Customizing+Columns

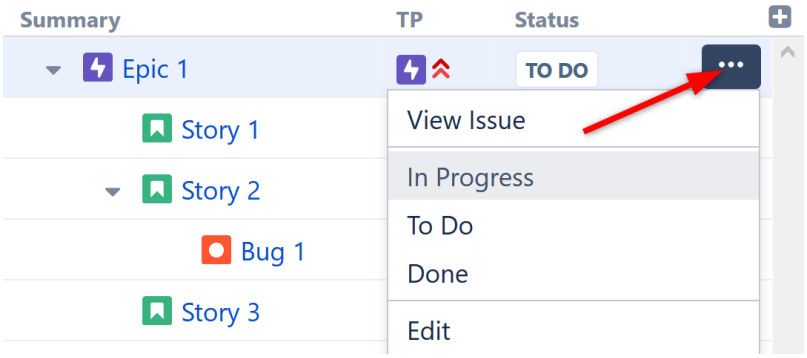
3.10.1.10 Jira Actions Column

The Jira Actions column allows you to open a menu with available Jira actions for the issue. This column works like the similar column on the Jira's Issue Navigator page and lets you log work, apply workflow actions and use other Jira actions(see page 139) available for the issue.



To use the Jira Actions menu:

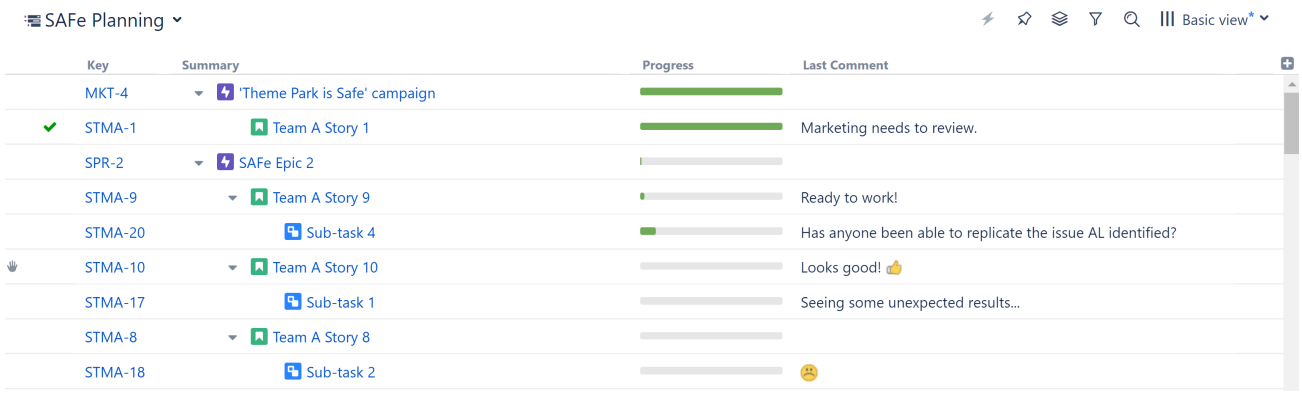
- Click the **...** button and select the desired action with the mouse, or
- Use the keyboard shortcut **Alt+Down Arrow** to open the menu for the currently selected issue and then use the **Up** and **Down** arrow keys and **Enter** key to select the action



> ⚠ The **...** button is only visible when the row is selected or when you hover over the Jira Actions column with the mouse.

3.10.1.11 Last Comment Column

The Last Comment column displays the last comment left for each issue.



Adding this column provides a simple way to quickly review the most recent comments or questions left on every issue in the structure.

Permissions

Users will see the last comment they have permission to view. If a users does not have permission to see the latest comment, Structure will display the most recent they do have permission to view.

Last Comments in Formulas and Effectors

The Last Comments attribute can also be used in:

- Formula Columns(see page 463) - filter calculations based on latest comments, or include the latest comments as part of your results
- Effectors(see page 179) - write recent comments to a custom field

3.10.1.12 Notes Column

The Notes column allows you to add arbitrary text to items in a structure, without having to create custom fields in Jira.

This is often used to store additional information about an item's status, which could then be used in a report - but how you use it is completely up to you and your business needs.

Portfolio Overview with Automation

Key	TP	Summary	Status	Notes
TP-124		Site preparations	OPEN	
SP-4		Flatten building site - Our theme park ne	OPEN	Bob's team
SP-2		Bulldoze French Alps - Someone p	OPEN	
SP-3		Remove waste rock and soil - After	IN PROGRESS	still need a confirmation
SP-6		Build access road - We need an eig	IN PROGRESS	the attachments need improvement
SP-11		Build access and protection	OPEN	
SP-9		Build a transparent dome over the th	IN PROGRESS	is it necessary?
SP-15		Install entrance checkpoints	OPEN	
SP-8		Dig escape tunnels for staff - We n	OPEN	should be closed. see the report
SP-6		Build access road - We need an eight-la	IN PROGRESS	the attachments need improvement
SP-10		Move stuff to another place	OPEN	consider teleportation
TP-31		Marketing + PR activities	OPEN	
MKT-4		'Theme Park is Safe' campaign - We nee	TO DO	check who has the same slogan
TP-8		Rides + attractions	OPEN	pitch new ideas to Mark please

The values in the Notes columns are **per-structure, per-item**. This means that:

- Text entered as Notes for some issue in one structure will not be seen for that issue in another structure. You may have different notes for the same issue in different structures.
- If an item occurs several times in a structure, they will have the same value in the Notes column.

The Notes column is great for leaving issue-level notes. For project-level notes, consider adding a Memo item(see page 105) to the structure.

Permissions

Data stored in the Notes column is considered to be a property of the selected structure. That has the following effect on the permissions.

Who can view notes?

To be able to see the notes, the user needs to have:

- View access to the structure that stores the notes.
- View access to the item (issue, project, etc.)

Who can edit notes?

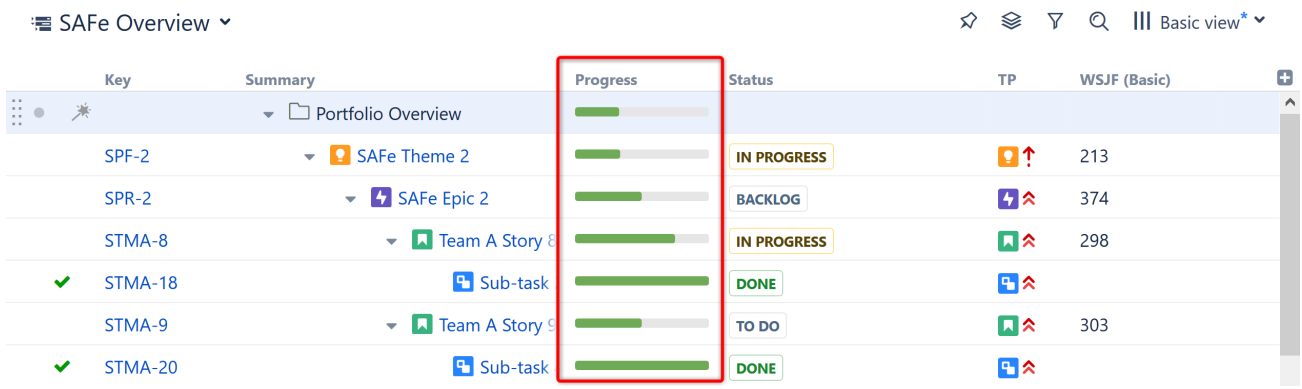
To be able to edit the notes, the user needs to have:

- Edit access to the structure that stores the notes.
- View access to the item (issue, project, etc.)

A user might have permission to edit notes, even if he or she does not have permission to edit the issue. By creating their own structure, a user can leave notes for issues they can't edit.

3.10.1.13 Progress Column

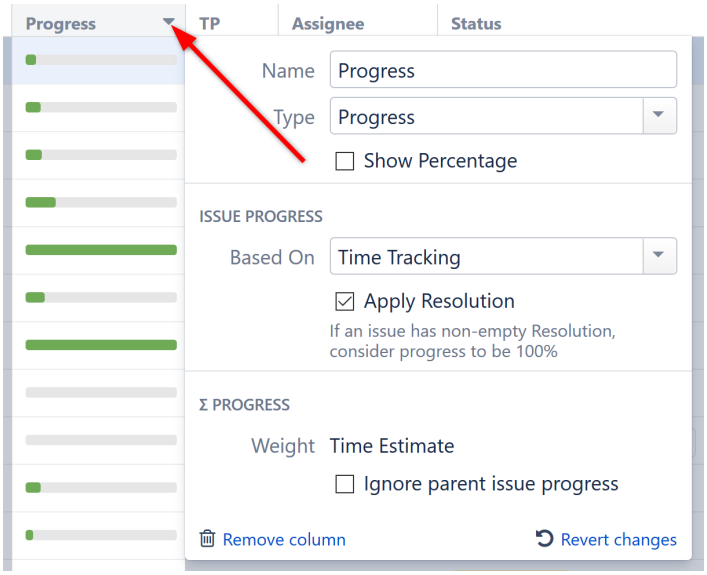
The Progress column displays an aggregate issue progress, which is calculated based on values from the issue and its sub-issues.



Progress can be based on:

- Time tracking,(see page 473)
- Resolution,(see page 477)
- Status,(see page 479) or
- A custom percentage field(see page 482).

You can select which type of progress column to include using the Add Column(see page 501) menu. Once a progress column has been added to your structure, you can change how it's calculated or customize its configuration using the column configuration panel(see page 502).



Progress is a custom Structure column, not available in the Issue Navigator or other standard Jira views.

How is Progress Calculated?

Progress calculations can be customized in two parts:

1. How the individual issue progress is calculated, regardless of its position in the structure.
2. How the progress of sub-issues are aggregated and combined with the individual progress of the parent issue.

Individual Issue Progress Calculation

There are several progress calculation modes. You can select which one to use in the **Based On** option:

- Progress Based on Time Tracking(see page 473) — The progress is calculated based on the issue's Resolution field, time tracking data and the progress of its sub-issues.
- Progress Based on Resolution Only(see page 477) — The progress is calculated based on the issue's Resolution field and the progress of its sub-issues.
- Progress Based on Status(see page 479) — The progress is determined by the issue's Status field. Custom percentage values can be assigned to each status.
- Progress Based on Percent Field(see page 482) — The progress is assigned to each issue manually in a custom field, and progress is aggregated for parent issues.

Total Progress Calculation

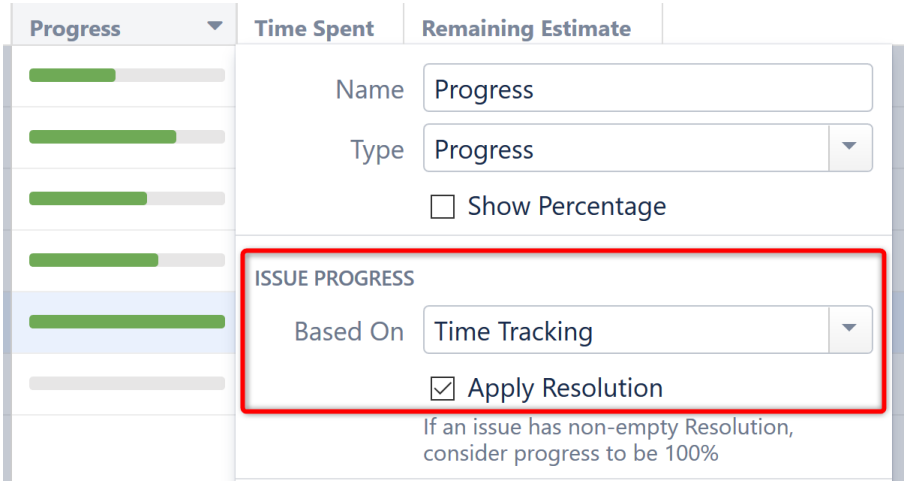
When individual issue progress is calculated based on Status(see page 479), Percent Field(see page 482), or Resolution Only(see page 477), you can specify how sub-issue progresses are aggregated into their parent issue progress. This is defined by the **Weight** option:

- **All Sub-Issues Are Equal** – All sub-issues are considered equal when calculating aggregated progress for the parent issue. Weights do not accumulate, so sub-issues of each level are considered equal, irrespective of how many sub-sub-issues they have.
- **Time Estimate** – Sub-issues' progresses are weighted proportionally to their total time estimate (*Time Spent + Remaining Estimate*). This option is akin to **Time Tracking**, but allows you to get individual progress from other sources (such as a numeric custom field or the Status field). If time information is not present, it is counted in as an average, based on the mean total time (time spent + remaining estimate) across sub-issues.
- **Numeric Field** - Sub-issues' progresses can be weighted proportionally based on a selected numeric field, such as Story Points or other custom field. Weights are accumulated upwards. If the field value is not present, it is counted as an average, based on the mean field value across sub-issues.

A zero value in the field configured as weight will discard any issue's progress in the parent issue aggregation.

Progress Based on Time Tracking

When Issue Progress is based on Time Tracking within a Progress column, the progress is calculated based on the issue's Resolution field, time tracking data, and the progress of its sub-issues.



Calculating Progress for Issue Without Sub-Issues

If the issue does not have sub-issues:

- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100%.
- Otherwise, if the issue has time tracking information, the progress is calculated proportionally to this issue completion %: `(Time Spent) / (Time Spent + Remaining Estimate)`
- Otherwise, the progress is 0%.

Calculating Progress for Issue with Sub-Issues

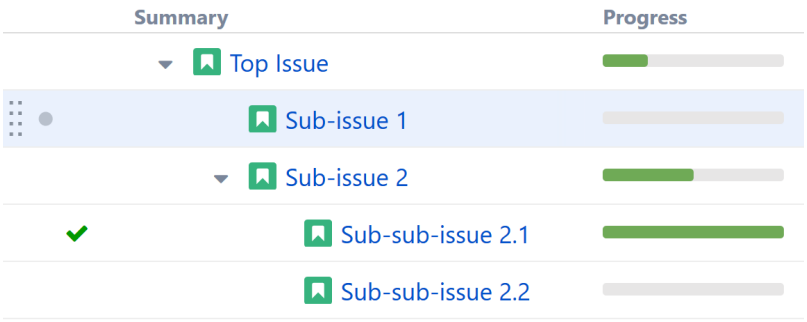
If the issue has sub-issues:

- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100% - regardless of the sub-issues' progress.
- If the issue and its sub-issues do not have estimates or work logged (or if time tracking is turned off), the progress is calculated as the average of the sub-issues' progresses.
- If time tracking is used and all issues have an estimate (either original estimate or remaining estimate), the estimates and total work logged are summed up and the progress is calculated as the total completion %: `(Total Time Spent) / (Total Time Spent + Total Remaining Estimate)`
    - If a sub-issue does not have time tracking information, it is counted in as an average sub-issue, based on the mean total time (time spent + remaining estimate) of its siblings.

If the issue has both its own time tracking information and sub-issues with progress, and if **Ignore Parent Issue Progress** is turned off, the issue's own progress value is counted as if it was the progress of another sub-issue.

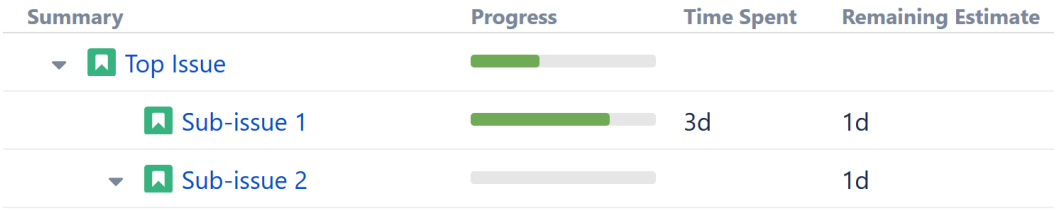
Examples

1. Without Time Estimates



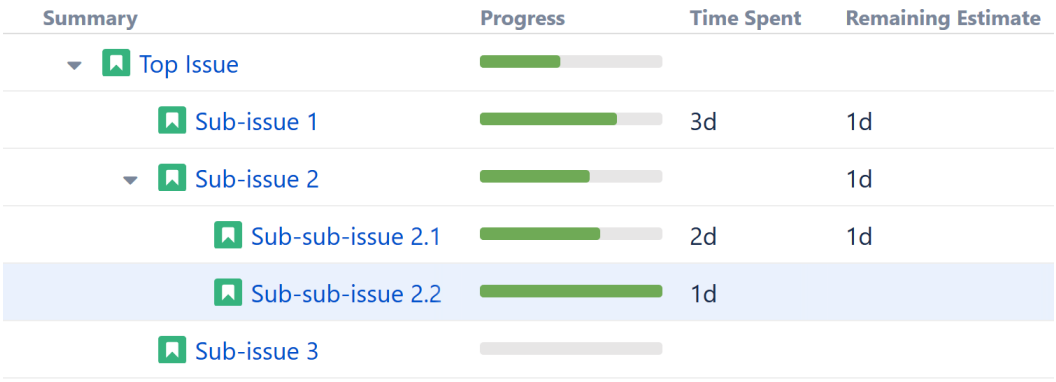
Issue	Explanation	Progress
Sub-sub-issue 2.1	The issue is resolved (indicated by the green mark), so it is complete.	100%
Sub-issue 2	It has two sub-issues with 100% and 0% progress; the total progress is the average of the two.	50%
Top issue	It has two Sub-issues: sub-issue 1 is 0% done and Sub-issue 2 is 50% done; the mean value is 25%.	25%

2. With Time Tracking Information



Issue	Explanation	Progress
Sub-issue 1	It has 3 days of work logged with 1 day remaining, so its progress is `time spent / total time = 3 / (3 + 1)`.	75%
Sub-issue 2	This issue does not have any work logged, is not resolved and does not have sub-issues.	0%
Top issue	The top issue has a total time spent of 3 days (work logged on Sub-issue 1) and 2 total days remaining (estimates on Sub-issue 1 and Sub-issue 2), so its progress `3 / (3 + 2)`.	60%

3. More Complex Example

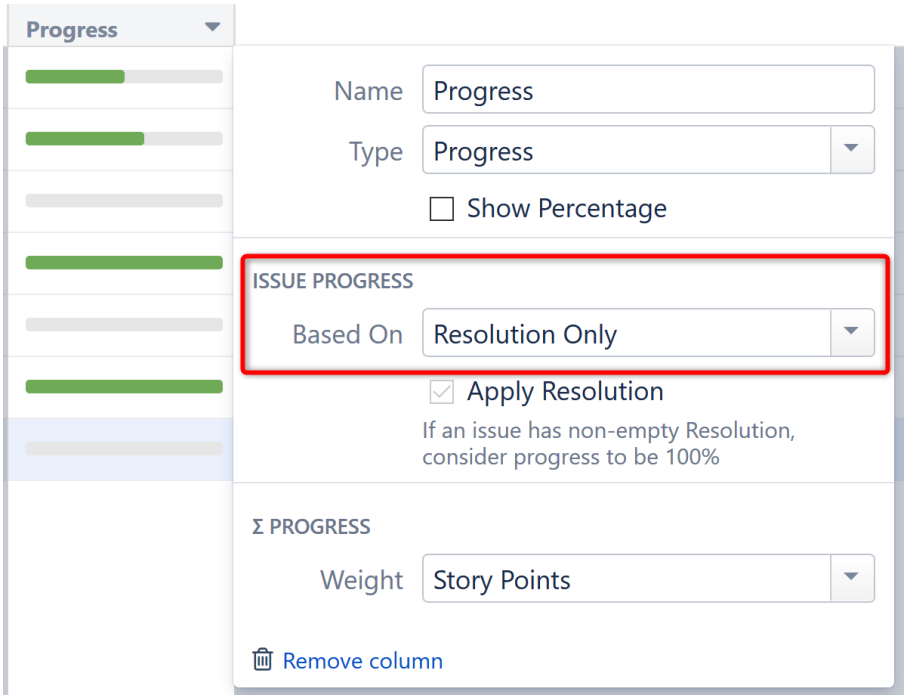


Issue	Explanation	Progress
Sub-sub-issue 2.1	It has 2 days of work logged with 1 day remaining, so its progress is `2 / (2 + 1)`.	66%
Sub-sub-issue 2.2	This issue has 1 day of work logged and no work remaining - so even though it is not resolved, it's considered completed.	100%
Sub-issue 2	It has total time spent of 3 days, and total remaining estimate of 2 days (the remaining time from Sub-sub-issue 2.1 and its own 1 day, which is considered additional work). The progress is `3 / (3 + 2)`.	60%
Sub-issue 1	This one has 3 days of work logged and 1 day remaining, so its progress is `3 / (3 + 1)`.	75%

Issue	Explanation	Progress
Top issue	The obvious total time spent is 6 days with a total remaining estimate of 3 days (the count from all sub-issues on all levels). But there's also *Sub-issue 3*, which does not have any estimates or work logged, so it gets estimated based on the average of its siblings - *Sub-issue 1* and *Sub-issue 2*. The progress of the top issue is calculated as follows: • The average between the total time of *Sub-issue 1* (3 + 1 = 4 days) and the total time of *Sub-issue 2* (3 + 2 = 5 days) is 4.5 days. So *sub-issue 3* is treated as if it has a total time of 4.5 days. • Since Sub-Issue 3 has 0% progress (because there is no time logged), it is also treated as if it has a remaining estimate of 4.5 days. • Top Issue is then calculated as having a total time spent of 6 days and total remaining time of 7.5 days, so its progress is calculated as `6 / (6 + 7.5)`.	44%

Progress Based on Resolution Only

When Issue Progress is based on Resolution Only within a Progress column, the progress is calculated based on the issue's Resolution field and the progress of its sub-issues.



Calculating Progress for Issue Without Sub-Issues

If the issue does not have sub-issues:

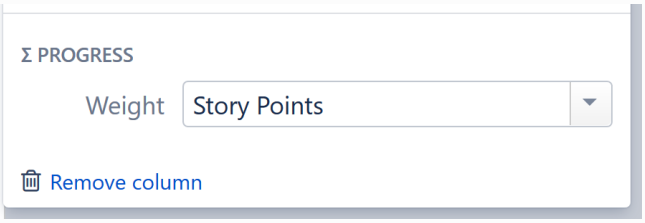
- If the issue's Resolution field is not empty, the progress is 100%.
- Otherwise, the progress is 0%.

Calculating Progress for Issue with Sub-Issues

If the issue does have sub-issues:

- If the issue's Resolution field is not empty, the progress is 100% - regardless of the sub-issues' progress.
- Otherwise, the issue's progress is the weighted average of its sub-issues.

You can specify how an issue's weight is determined in the column configuration panel(see page 502).



Example

Resolution Only with Story Points
When Issue Progress is based on Resolution Only and the Weight is determined by Story Points:

- Individual progress is 0% or 100% based on Resolution field
- Total progress is calculated as a weighted average, with weights contained in the *Story Points* field

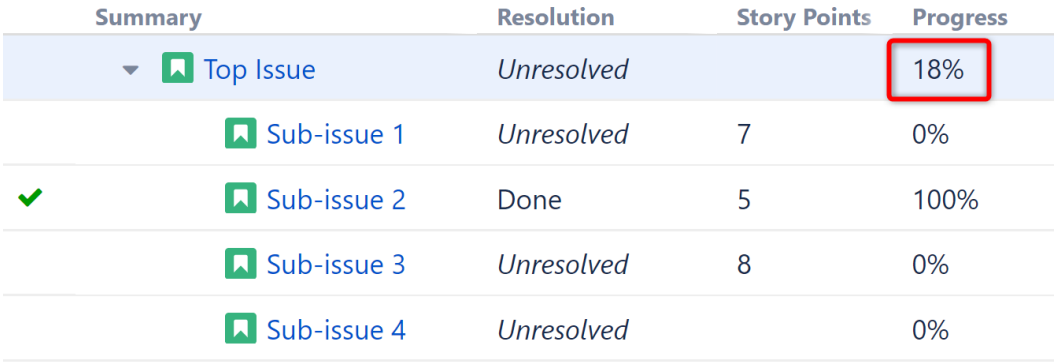
Summary	Resolution	Story Points	Progress
Top Issue	*Unresolved*		
Sub-issue 1	*Unresolved*		
Sub-sub-issue 2.1	*Unresolved*	2	
✔ Sub-sub-issue 2.2	Fixed	3	
Sub-issue 2	*Unresolved*	1	

Issue	Explanation	Progress
Sub-sub-issue 2.2	This issue is resolved (indicated by the green mark) - so it is complete.	100%

Issue	Explanation	Progress
Sub-issue 1	It has two sub-issues with 0% and 100% progress, and story points are 2 and 3 respectively. So the total progress is the weighted average value of (0 x 2 + 100 x 3) / (2 + 3).	60%
Top issue	It has two sub-issues: Sub-issue 1 (60% done) and Sub-issue 2 (0% done), and their cumulative story points are (2 + 3) and 1, respectively. So the progress is (60 x 5 + 0 x 1) / (5 + 1).	50%

Estimating Total Progress When Weight Values Are Missing

When calculating a parent's total progress, children without weight values are calculated using an average weight of their siblings.



Summary	Resolution	Story Points	Progress
Top Issue	*Unresolved*		18%
Sub-issue 1	*Unresolved*	7	0%
✔ Sub-issue 2	Done	5	100%
Sub-issue 3	*Unresolved*	8	0%
Sub-issue 4	*Unresolved*		0%

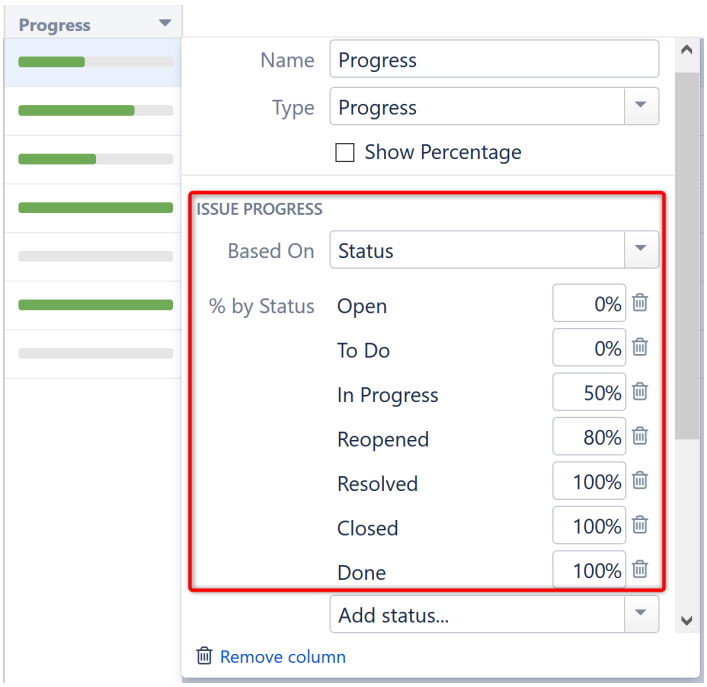
In the above example, we might assume that the progress of Top Issue would be 25%, because 5 story points are completed, out of total 20 (7+5+8) - but that only works if Sub-issue 4 requires no work to complete. That's probably not the case!

In order to account for the work Sub-issue 4 will require, Structure assumes an average number of story points (or whatever the selected weight value), based on the values of its siblings. In this case, that's the average of 7, 5, and 8 - or 6.66.

When Structure calculates the progress for Top Issue, it compares the completed story points of all siblings (5) against the combined actual and estimated story points (7+5+8+6.66), for a total progress of 18%.

Progress Based on Status

When Issue Progress is based on Status within a Progress column, the progress is determined by the issue's Status field. Custom percentage values can be assigned to each status.



Calculating Progress for Issue Without Sub-Issues

If the issue does not have sub-issues:

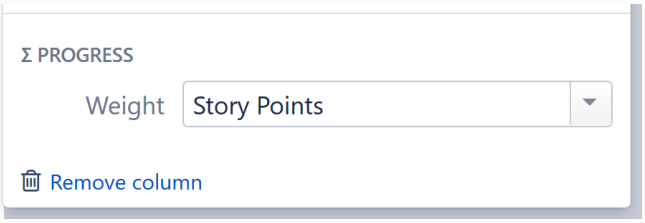
- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100%.
- If the issue's Status is assigned a value (%) in the column configuration, the progress is equal to that value.
- Otherwise, the progress is undefined. The issue neither shows any progress, nor affects the progress of its parent issue.

Calculating Progress for Issue with Sub-Issues

If the issue does have sub-issues:

- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100% - regardless of the sub-issues' progress.
- Otherwise, the issue's progress is the weighted average of its sub-issues.
    - If the issue has both its own status and sub-issues with progress, and if **Ignore Parent Issue Progress** is turned off, the issue's own progress value is counted as if was the progress of another sub-issue.

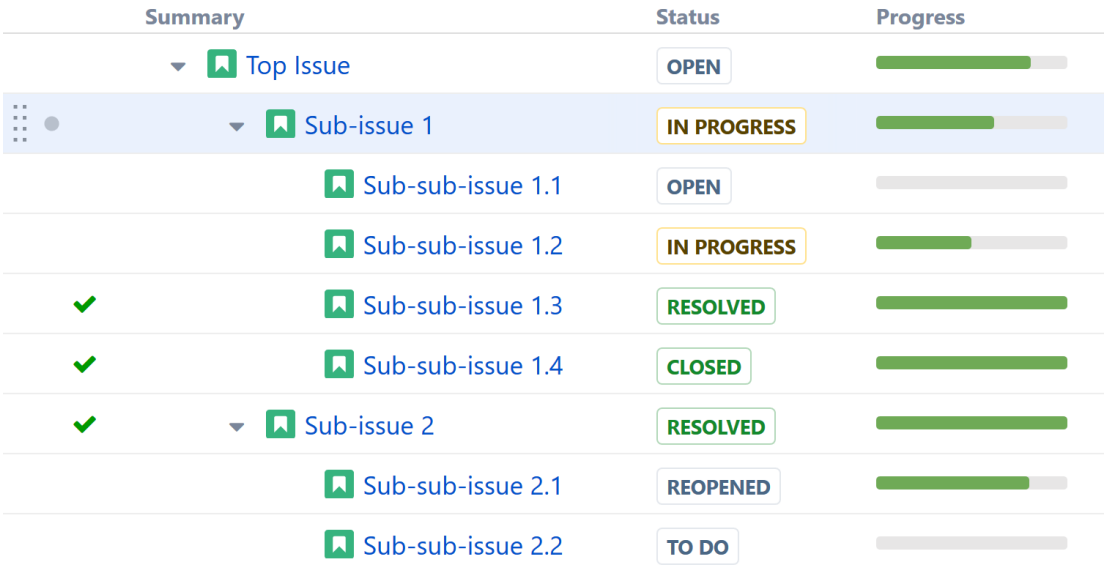
You can specify how an issue's weight is determined in the column configuration panel(see page 502).



If some of the statuses do not have any percentage configured, the issue progress is considered undefined.Issues with those statuses will neither show any progress, nor affect the progress of their parent issues.

Example

Progress Based on Status, All Sub-Issues Are Equal
In this example, statuses have the following percentages: Open = 0%, In Progress = 50%, Resolved or Closed = 100%, Reopened = 80%. **Apply Resolution** is turned on, and **Ignore Parent Issue Progress** is turned on.

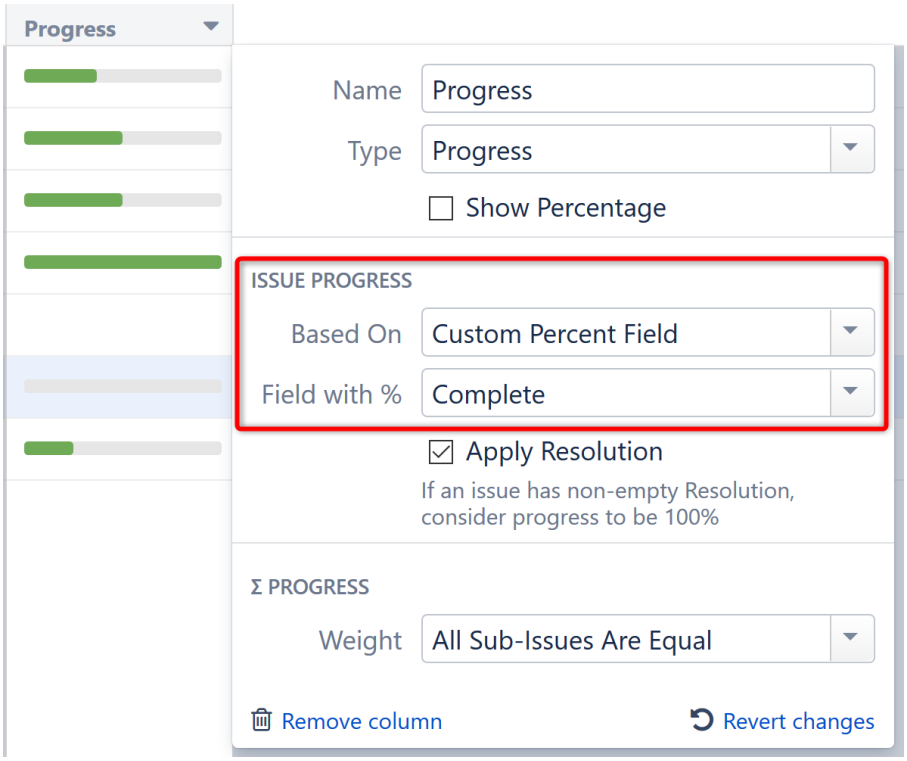


Issue	Explanation	Progress
Sub-sub-issue 1.1	This issue is Open, so the progress is 0%.	0%
Sub-sub-issue 1.2	This issue is In Progress, so progress is 50%.	50%
Sub-sub-issue 1.3	This issue is Resolved, so progress is 100%. Also, according to the workflow, it has a non-empty Resolution, which also means it's complete.	100%
Sub-sub-issue 1.4	This issue is Closed, so progress is 100%. Also, according to workflow, it has a non-empty Resolution, which also means it's complete.	100%

Issue	Explanation	Progress
Sub-issue 1	The average progress of its sub-issues is (0+50+100+100)/4. The issue's own status is In Progress, but it's percentage is ignored because of the "Ignore parent issue progress in aggregation" option.	63%
Sub-sub-issue 2.1	This issue is Reopened, so progress is 80%.	80%
Sub-sub-issue 2.2	This issue is Open, so progress is 0%.	0%
Sub-issue 2	The average progress of its sub-issues is (80+0)/2 = 40%. But the issue itself has a Resolution and the "Issues with Resolution are 100% done" option is turned on, so this overrides the sub-issues' progress and makes the issue complete.	100%
Top issue	It has two sub-issues: Sub-issue 1 is 63% done and Sub-issue 2 is 100% done. Average progress is (63+100)/2.	82%

Progress Based on Percent Field

When Issue Progress is based on a custom percent field, the progress is assigned to each issue manually in a custom field, and progress is aggregated for parent issues.



You can use any numeric Jira custom field to store the current progress % – a value from 0 to 100.

Calculating Progress for Issue Without Sub-Issues

If the issue does not have sub-issues:

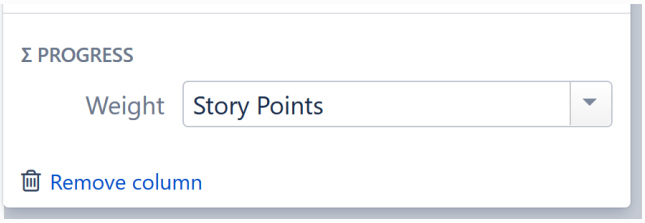
- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100%.
- If the issue's Custom Field value is not empty and is between 0 and 100, it's considered as the completion progress in percents.
- If the issue's Custom Field value is less than 0, the progress is 0%; if greater than 100, the progress is 100%.
- Otherwise, the progress is undefined. The issue neither shows any progress, nor affects the progress of its parent issue.

Calculating Progress for Issue with Sub-Issues

If the issue does have sub-issues:

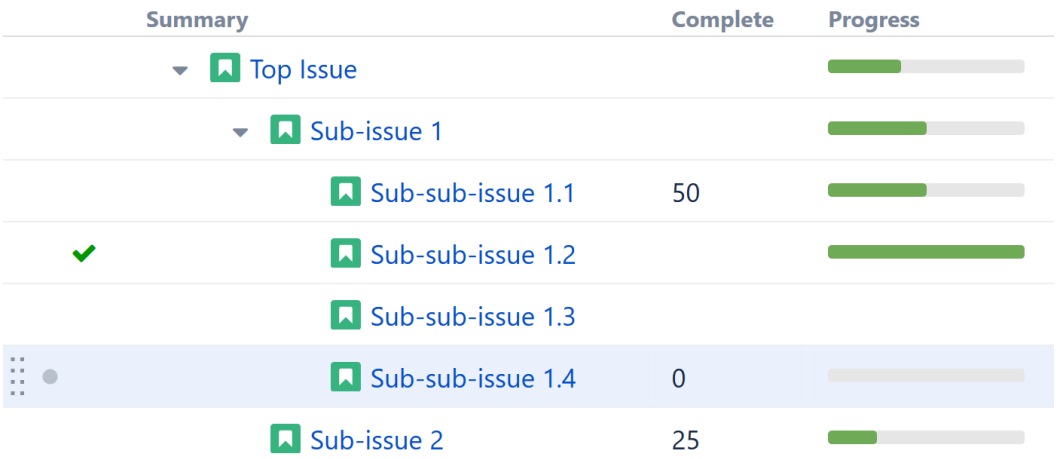
- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100% – regardless of the sub-issues' progress.
- If the issue's Custom Field value is not empty, it's considered as that issue's completion progress in percents (from 0 to 100) – regardless of the sub-issues' progress.
- Otherwise, the issue's progress is the weighted average of its sub-issues.

You can specify how an issue's weight is determined in the column configuration panel(see page 502).



Examples

1. Percent Field, All Sub-Issues Are Equal
With a Custom field named *Complete*, total progress based on **All Sub-Issues Are Equal**, and **Apply Resolution** turned on.



Issue	Explanation	Progress
Sub-sub-issue 1.1	This issue is 50% complete as specified by the custom field.	50%
Sub-sub-issue 1.2	This issue is resolved (indicated by the green mark) - so it is complete, even if the "Complete" field is empty.	100%
Sub-sub-issue 1.3	This issue has no progress information (neither "Resolution" nor "Complete" fields), so progress is undefined and not counted at all.	n/a

Issue	Explanation	Progress
Sub-sub-issue 1.4	This issue has a 0 "Complete" value, which means it's 0% complete.	0%
Sub-issue 1	It has four sub-issues, but 1.3 is ignored. So the total progress is the average of the rest: (50 + 100 + 0) / 3.	50%
Sub-issue 2	The issue is 25% complete as specified by the custom field.	25%
Top issue	It has two sub-issues: Sub-issue 1 is 50% done and Sub-issue 2 is 25% done. So the progress is the average of the two: (25 + 50) / 2.	38%

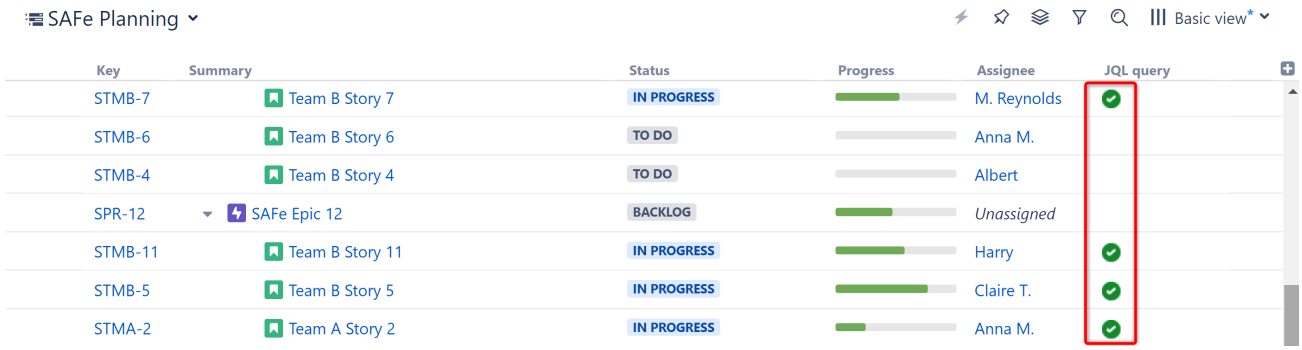
2. Percent Field, Story Points
With a Custom field named *Complete*, total progress based on the field *Story Points*, and **Apply Resolution** turned on.

Issue	Explanation	Progress
Sub-sub-issue 1.1	This issue is 50% complete as specified by the custom field.	50%
Sub-sub-issue 1.2	This issue is resolved (indicated by the green mark) - so it is complete, even if the "Complete" field is empty.	100%

Issue	Explanation	Progress
Sub-sub-issue 1.3	This issue has no progress information (neither "Resolution" nor "Complete" fields), so progress is undefined and not counted at all.	n/a
Sub-sub-issue 1.4	This issue has a 0 "Complete" value, which means it's 0% complete.	0%
Sub-issue 1	It has four sub-issues, and weight is based on story points: • 1.1 has 2 story points • 1.2 has 3 story points • 1.3 is ignored (because its progress is undefined) • 1.4 has no story points, so it's weight is calculated as the mean of its siblings (2 and 3 = 2.5) The total progress is the weighted average of 1.1, 1.2 and 1.4: (50 x 2 + 100 x 3 + 0 x 2.5) / (2 + 3 + 2.5).	53%
Sub-issue 2	The issue is 25% complete as specified by the custom field.	25%
Top issue	It has two sub-issues: Sub-issue 1 is 53% done and has 7.5 story points; Sub-issue 2 is 25% done and has 1 story point. So the progress is calculated as (53 x 7.5 + 25 x 1) / (7.5 + 1).	50%

3.10.1.14 Query Match Column

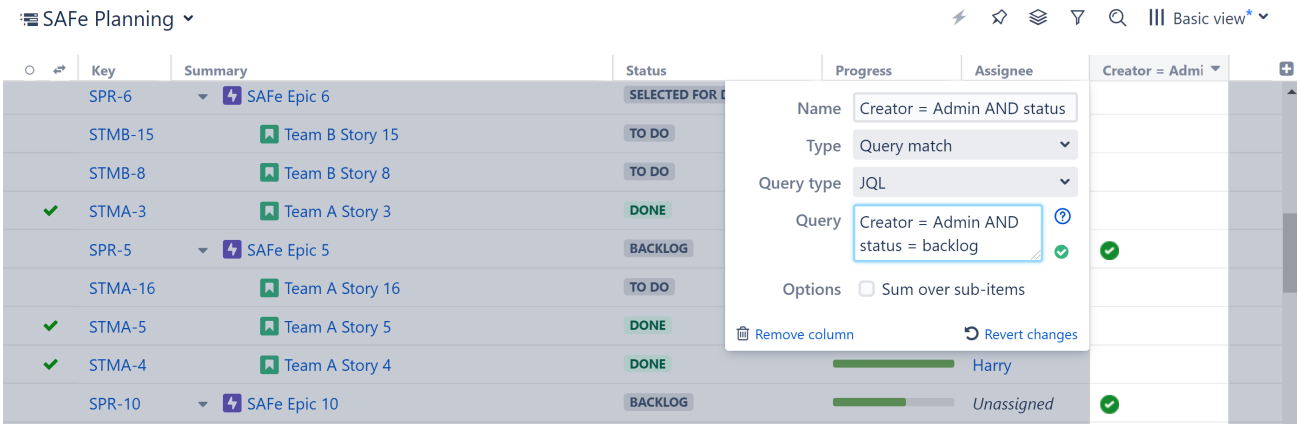
The Query Match column allows you to easily identify issues that match a specified text, JQL or S-JQL(see page 428) query. Items that match the query are noted by a check mark; those that do not are left blank.



Adding a Query Match Column

To add a Query Match column to a structure, click the **+** icon to the right of the column headers. In the Add Column dialogue, search for **Query match...**

In the Column Configuration panel, select the type of query (text, JQL or S-JQL(see page 428)) and enter your query. The name will update automatically based on your query, but you can also enter a custom name.

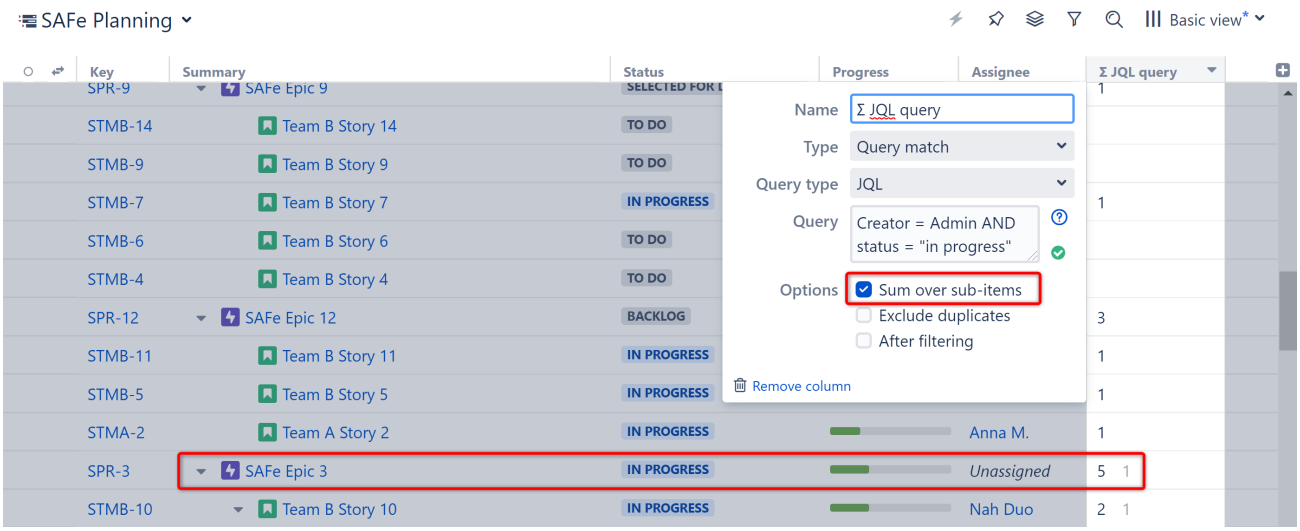


In our example, the Query match shows us any issues that were created by Admin and are still in the backlog.

> Query match can also be used in formulas(see page 229).

Sum over sub-items

The Query Match column can also return a numeric value of "1" for any issue matching the query, when you select **S um over sub-items**. This allows you to aggregate your results up the hierarchy.



In this example, Safe Epic 3 has 5 items (including itself) that were created by Admin and are in progress.

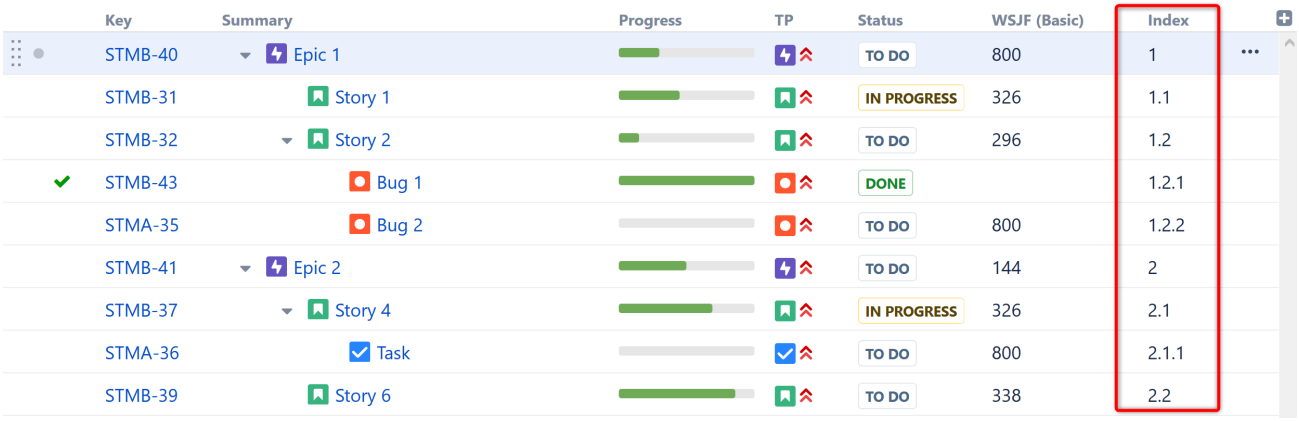
Aggregation Options

When **Sum over sub-items** is selected, you have a couple of options:

- **Exclude duplicates** - If an item appears more than once in the structure, it's value will only be included once within the aggregate total.
- **After filtering** - When checked, filtered items will not be included in the aggregate total. If this is left unchecked, the values of those items will be included in the calculation, even though they are not visible in the structure.

3.10.1.15 Sequential Index

The Sequential Index column displays a hierarchical number corresponding to an item's position within the structure.



The first number in the index represents the top level of the hierarchy, the next number (after the period) refers to the second level, etc. To better understand how this works, let's look at the above example:

- **Epic 1** has a sequential index of 1, because it is the first item in the top level of the hierarchy.
- **Story 2** has a sequential index of 1.2, because it is the second child under **Epic 1**.

- **Bug 1** has a sequential index of 1.2.1, because it is the first child under **Story 2**.
- **Task** has a sequential index of 2.1.1, because **Epic 2** is second item in top level, **Story 4** is the first item beneath it, and **Task** is the first child beneath that.

Sequential Index can also be used in Formulas(see page 228).

Sequential Index and Filter Transformations

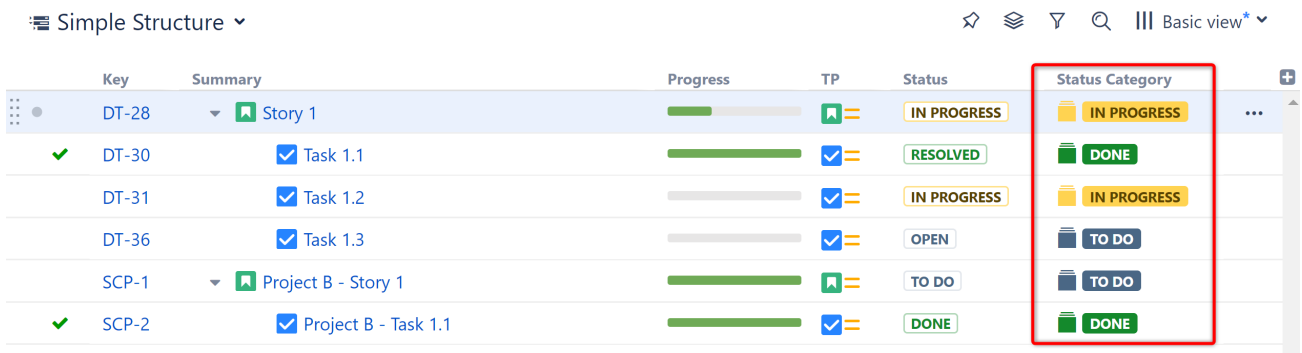
The Sequential Index attribute works different depending on whether you're viewing it in a column or using it in a formula:

- The Sequential Index column ignores Filter Transformations(see page 207), so even if you see only a part of a structure, the numbers will still show the position of the item in the unfiltered structure.
- When you use Sequent Index in a formula, Filter Transformations are taken into consideration and will affect your final calculation.

Filter generators(see page 158) always affect the Sequential Index because they change the underlying structure.

3.10.1.16 Status Category Column

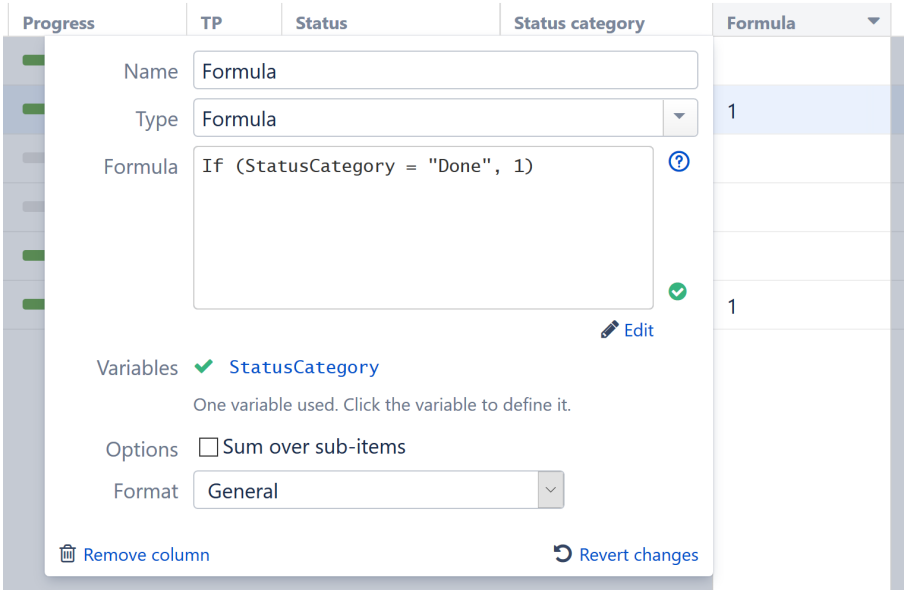
This column allows you to see at a glance which Status category each issue is in. This can be extremely useful if different teams/projects use different workflows or custom statuses.



Status category is a read-only column.

Status Category Values

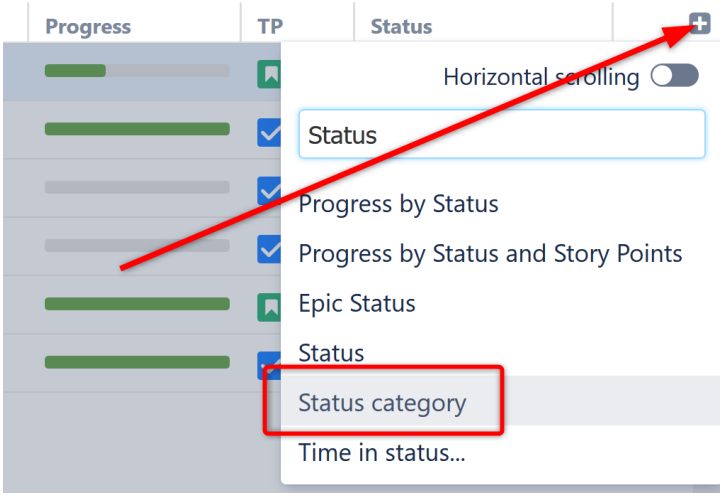
Status category values can be used to sort, filter or group issues. They can also be used in JQL queries and formulas(see page 228).



To group issues by Status category, use Group by Text Attribute.

Adding the Status Category Column

To add the Status category column to a structure, open the Add Column menu and search for **Status category**.

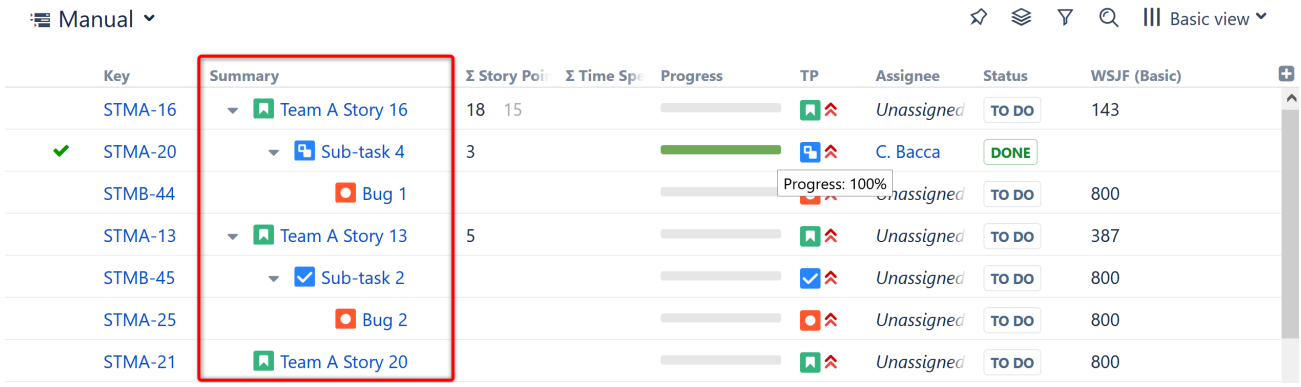


3.10.1.17 Summary Column

The Summary column displays the issue summary and, optionally, part of the issue description. For folders, it shows the folder name.
Summary can be edited right in the structure widget(see page 132) and it is the only field required for creating new issues(see page 128).

Structure Hierarchy

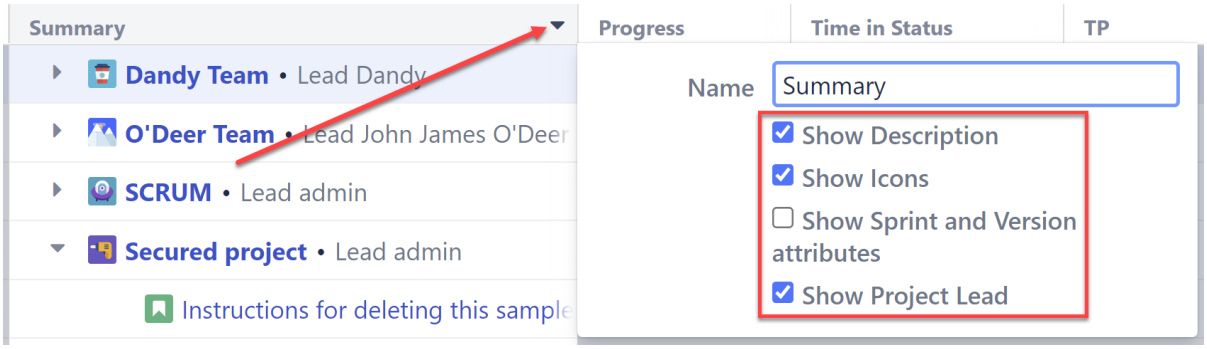
The Summary column is also where the structure hierarchy is displayed. The text for sub-items is indented relative to their parent items.



Customizing the Summary Column

To customize the Summary column, click the arrow icon in the column header and select which items should be displayed:

- Item Description
- Item Icons
- Sprint and Version attributes (Sprint start date, end date, and sprint goal; Fix Version start date, release date, and description) - only applies when items are grouped by Sprint or Version
- Show Project Lead - only applies when items are grouped by Project



The Summary column cannot be removed from the Structure grid or reconfigured to a different column type, because it displays the hierarchy.

3.10.1.18 Tempo Work Logged Column

The Tempo Work Logged column lets you view and manage Timesheets by Tempo work logged data within a structure:

- Track logged and billable time alongside your project data

- Aggregate the time spent across any project hierarchy
- Create multiple columns to track specific users, time frames, billable/non-billable time, and more

Key	Summary	All Work	Billable Work	Progress	Status
BP-3	Main Epic	1w 1d 1h	1w 4h		IN PROGRESS
BP-2	Issue 1	1d 7h	1d 5h		DONE
BP-4	Issue 2	1d 6h	1d 3h		DONE
BP-5	Issue 3				TO DO
BP-1	Issue 4	2d 4h	2d 4h		TO DO
BP-12	Issue 10				TO DO
BP-11	Secondary Epic	2w 4d 4h	2w 2d 6h 30m		IN PROGRESS
BP-8	Story 7	7h	4h		TO DO
BP-6	Story 5	4d 5h	4d 1h 30m		IN PROGRESS
BP-7	Story 6	4d	3d 6h		IN PROGRESS
BP-9	Story 8	3d	2d 4h		IN PROGRESS
BP-10	Story 9	2d	1d 7h		IN PROGRESS

The structure will only include issues and work logged data the active user has access to, based on their permission settings in Jira, Structure, and Timesheets.

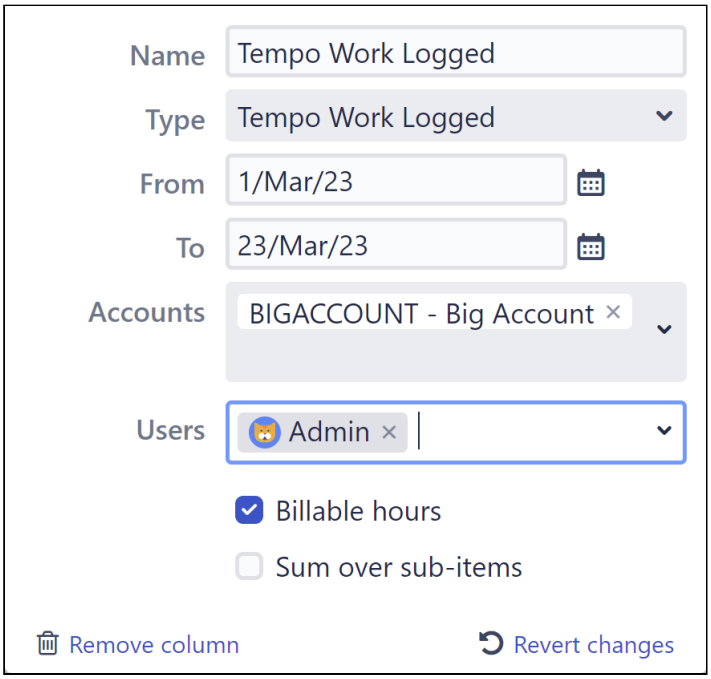
Grouping by Tempo Accounts and Teams

It is also possible to group items within a structure by Tempo Account and Tempo Teams. This can be useful if you want to track logged time by account or team. For more information, see: Timesheets by Tempo Integration(see page 600).

Customizing a Work Logged Column

You can customize each Tempo Work Logged column to focus on specific time frames, users, billable time, and more:

- **Name** - Give each Tempo Work Logged column a name specific to the data its displaying
- **Type** - Specifies the type of attribute used for the column (hint: it should always be "Tempo Work Logged")
- **From** - Specify a start date for the included work logged data
- **To** - Specify an end date for the included work logged data
- **Accounts** - Select which Tempo accounts you want to include (closed and archived accounts are not available)
- **Users** - Select the users you want to include (the time displayed will be the sum of all selected users)
- **Billiable hours** - When checked, only time marked Billiable will be included in the column total (when unchecked, all time will be included)
- **Sum over sub-items** - When checked, work logged values will be aggregated up your hierarchy



You can make as many Tempo Work Logged columns as you need. For example, you could add one column to display all work logged, and another to display just billable hours, so you can easily compare the two.

Troubleshooting

If you experience any problems adding or viewing a Tempo Work Logged column, refer to the following common troubleshooting tips.

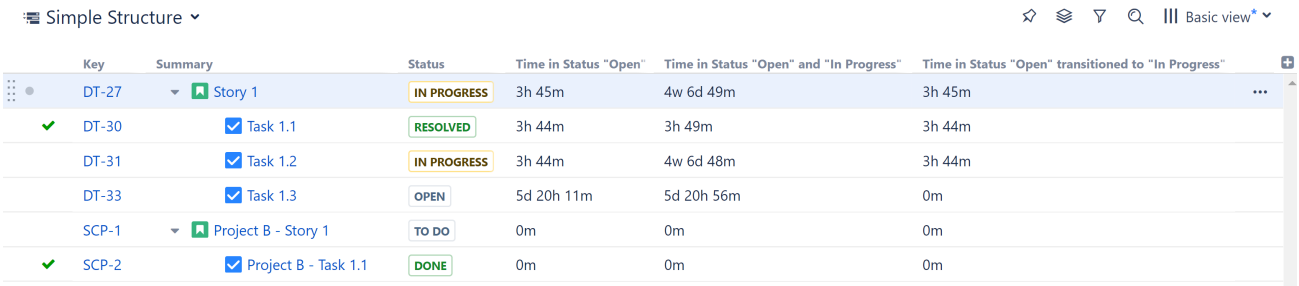
No data in the column - If you're not seeing any data in the column, or receiving an error message, check the following:

- Is work logged data available for the Period you've selected? Try adjusting the period and see if you get any results.
- Is there work logged data available for the Jira users you've selected, and do you have permission to view their work logged data? Try changing the selected users, or try selecting just yourself.

Data doesn't match what others see - If you're using a shared structure, the results in your Tempo Work Logged column my differ from someone else's, if you do not have the same permissions as the other person. For example, you may not have permission to view work logged data for everyone selected under Jira users.

3.10.1.19 Time in Status Column

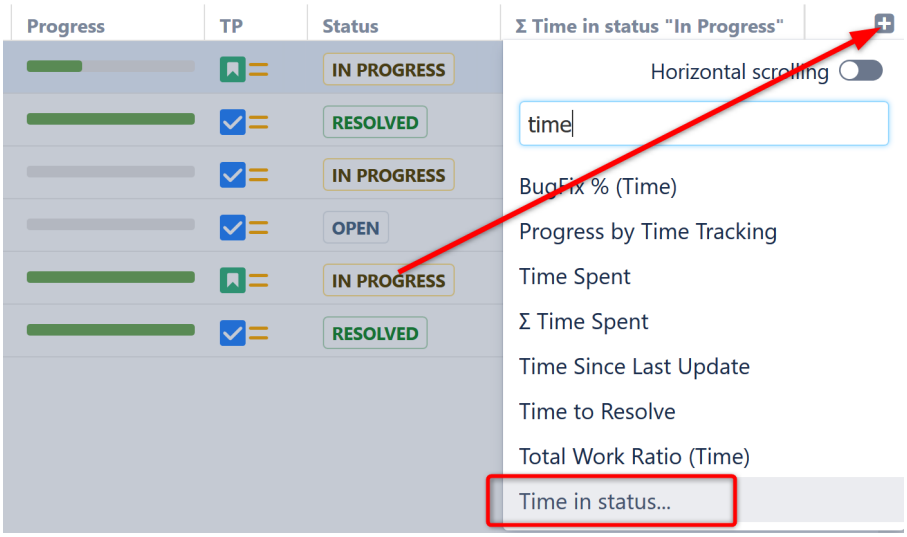
The Time in Status column allows you to calculate how much time issues spend in a particular status. It can also track multiple statuses, so you can see how much time issues spend in a particular part of the overall workflow.



Each Time in Status column (you can include as many as you need in a structure) can be customized to display precisely the data you need to see.

Adding a Time in Status Column

To add a new Time in Status column, open the Add Column menu and search for **Time in Status...**



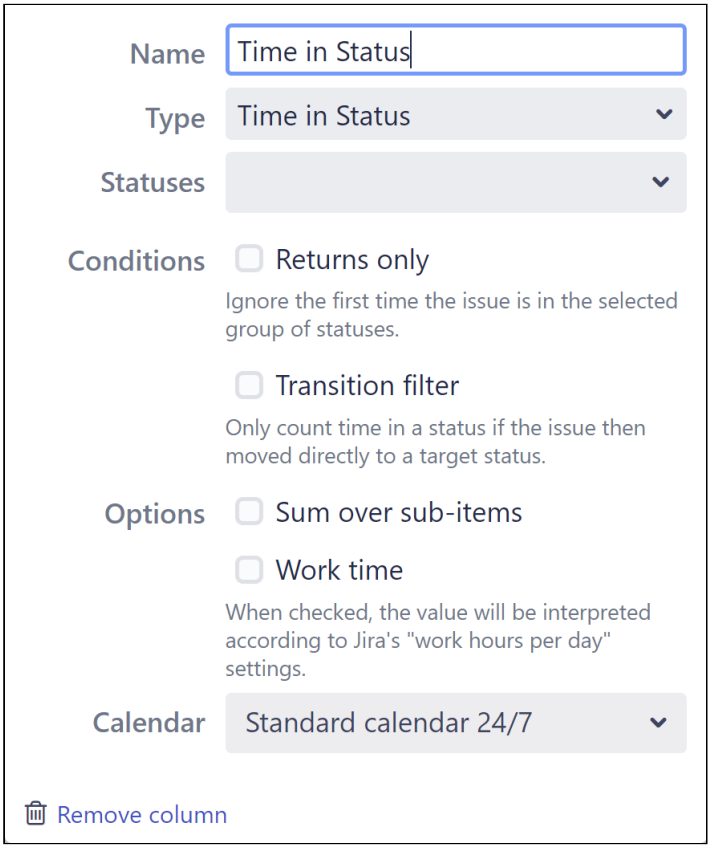
Once you've added the column to your structure, you can customize it to fit your specific needs.

Customizing a Time in Status Column

To customize a Time in Status column, open the column configuration panel (see page 502)and adjust any necessary settings:

- **Name** - we recommend giving the column a distinct name that alludes to the status(es) it's tracking, particularly if you have more than one Time in Status column in a structure.
- **Type** - this is the column type, and should remain unchanged, unless you wish to change an existing column to something else.
- **Statuses** - select which statuses and/or status categories to include when calculating time spent.
- **Returns only** - select this option to only track time when an issue returns to the selected statuses - see Returns only(see page 495) for full details.
- **Transition filter** - this option allows you to limit the results to only include time in a status when it directly preceded a move to a specific target status. For example, you could track just how long issues spend In Progress immediately before moving to Testing – if the issue moved from In Progress to any other status (back to Open, straight to Done, etc), that time would not be included.
- **Sum over sub-items** - select this option to get an aggregate total(see page 497) (all sub-issue values are summed up to their parent issue).

- **Work time** - when checked, the value is interpreted according to the Jira settings for work hours (work hours/day, work days/week).
- **Calendar** - choose a work calendar that should be used to calculate the time in status:
  - **Standard calendar 24/7** will consider all hours of every day.
  - **Standard work calendar 8/5** only considers time during a standard work week. For example, if a task has been in Open status for 6 days, using the 24/7 calendar it's time in status would be 6 days, with the 8/5 calendar this would be 1 week.
  - If you have Structure.Gantt installed, you can create custom calendars which can be selected here. For example, you could specify holidays and vacations, which would then be excluded from the time in status calculation.

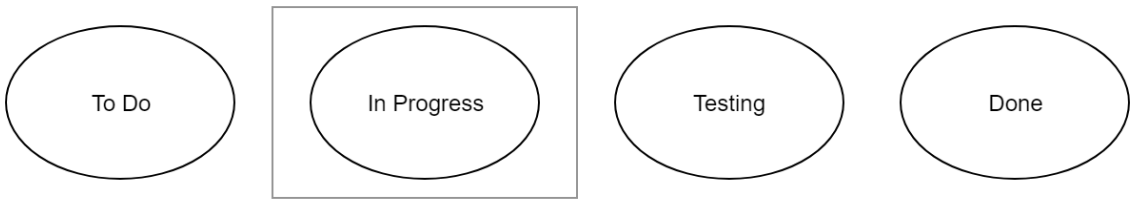


Returns only

When **Returns only** is selected, Structure will ignore the first time an issue is in the group of selected statuses/ status categories, and only track time if the issue returns to the selected statuses. This is a very useful way to track additional work.

Example 1 - A Single Selected Status

In the following workflow, you may want to see how much work has had to be re-done because issues are being passed along to testing too soon. To do so, under Statuses, you would select **In Progress**, and under Options check the **Returns only** box.

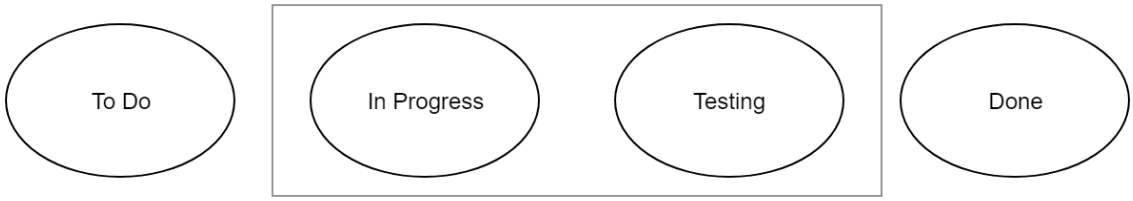


In this configuration, Structure would ignore the first time the issue was in the **In Progress** status, and only track time if it returned to the **In Progress** status after moving to Testing, Done, or back to To Do.

Example 2 - Multiple Selected Statuses

You may not have any concerns about items being returned to **In Progress** by your Testing teams, because that's just good practice - but you may be concerned with issues being returned to **In Progress** or **Testing** after they've been marked **Done**.

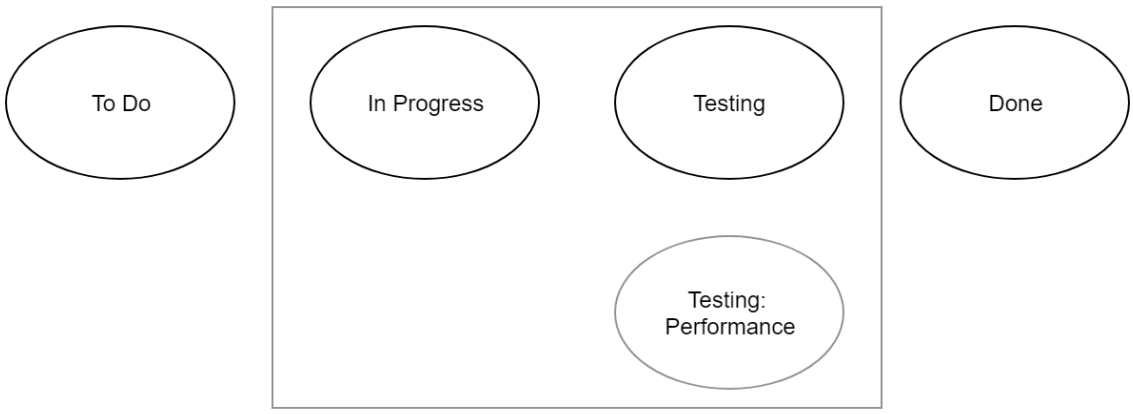
To do this, you could add both **In Progress** and **Testing** to your selected Statuses, and continue to select the **Returns only** box.



Now In Progress and Testing are considered a group to Structure - so moving from **In Progress** to **Testing** and then back to **In Progress** won't be considered a return (since the issue never left the group of selected statuses). But if the issue is moved to **Done** and then returns to either **In Progress** or **Testing**, that will be considered a return and Structure will track how much time is spent in those statuses after that point.

Example 3 - More Complex Setup

Many of us have multiple statuses for work in progress or testing. In this case, you can group all of those together (or just the relevant ones). In the example below, we've added **In Progress**, **Testing**, and **Testing: Performance** to our Statuses list.

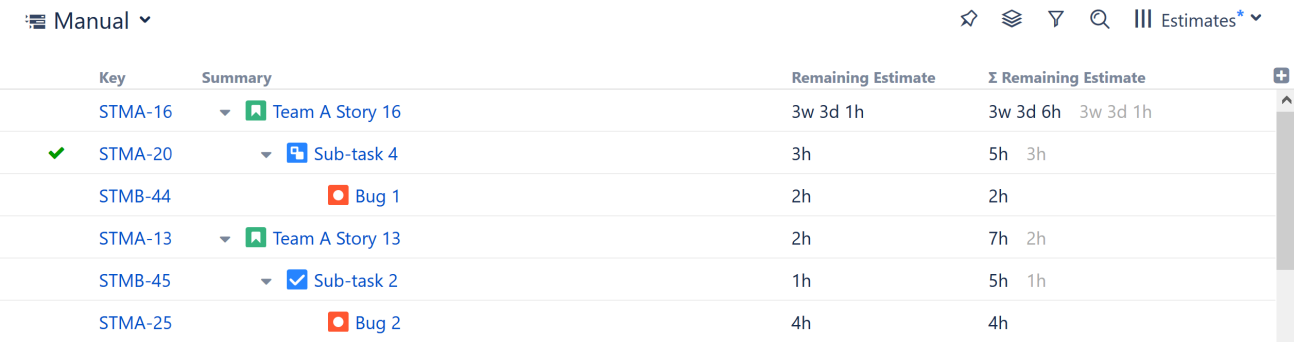


In this example, let's look at how Structure will calculate time if **Returns only** is checked:

- Issue moves from **To Do** → **In Progress** → **Testing** → **Testing: Performance** → **Done** - In this case, no time will be reported, because the issue never returned to a selected status.
- Issue moves from **To Do** → **In Progress** → **Testing** → **In Progress** → **Testing: Performance** → **Done** - Again, no time will be tracked. Even though the issue went back to **In Progress** a second time, it was continuously in one of the selected statuses, so this is not considered a return.
- Issue moves from **To Do** → **In Progress** → **Testing** → **Testing: Performance** → **Done** → **Testing: Performance** → **Done** - This time Structure will report the time spent the second time in **Testing: Performance**.
- Issue moves from **To Do** → **In Progress** → **Testing** → **Done** → **Testing: Performance** → **Done** - Structure will report the time spent in **Testing: Performance**. Even though the issue only reached this status once, it did so after leaving the selected statuses the first time; therefore, it is considered a return.

3.10.1.20 Totals Columns

Totals columns provide aggregate values for all numeric and time-tracking fields. These are calculated as the sum of the current item's field value and those of its sub-issues.

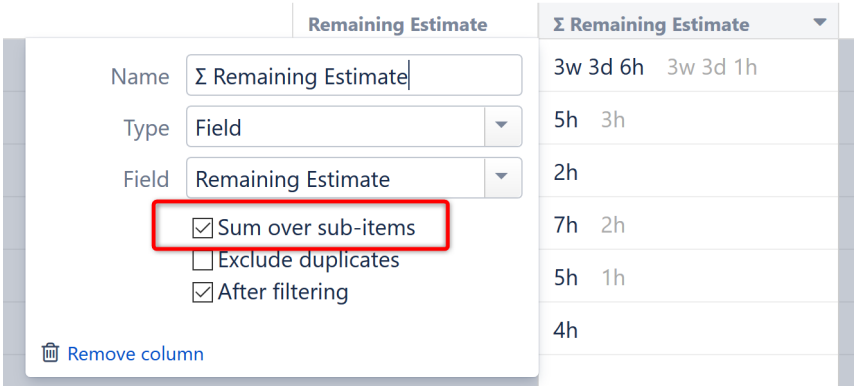


When using an aggregate column:

- When an aggregate value is displayed for an issue that also has its own value in the field, its own value is displayed next to the aggregate value in a gray color.
- Since issues can be present multiple times in a structure, you can select whether you want to count every instance of an issue or count it just once. By default, duplicates are counted each time they appear. To exclude them, select **Exclude Duplicates** in the column configuration panel(see page 0).

Sum over sub-items

All aggregate columns have the Sum over sub-items box checked.



This is what makes them aggregate columns. Unchecking this box will change the column from an aggregate column to it's corresponding Field column.

Build Your Own Aggregate Columns

You can also build your own aggregate columns for most numeric and time-tracking fields. Simply open the column configuration panel(see page 0) and select **Sum over sub-issues**.

If the Sum over sub-issues option is unavailable for a given field, then the aggregate cannot be calculated.

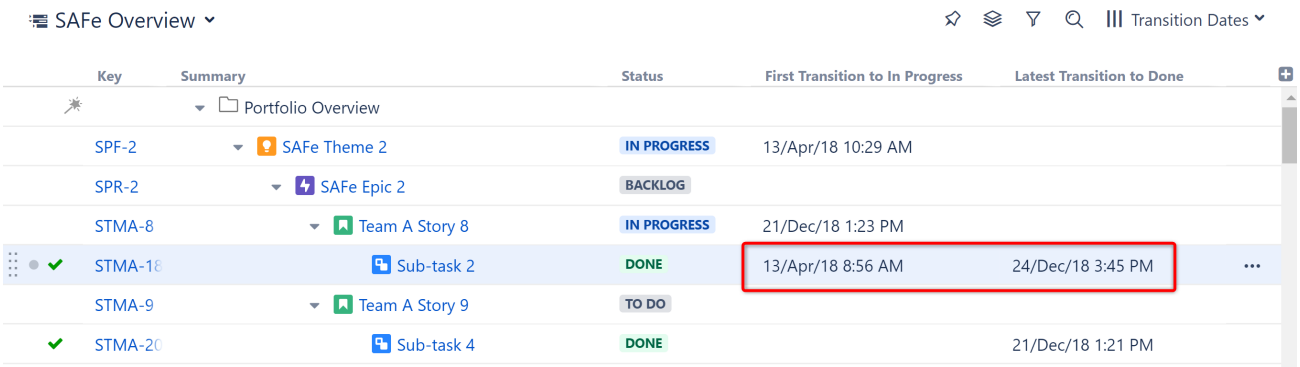
Totals are Structure-Specific

The Total value for a specific issue may change depending on the selected structure, because its sub-issues can vary by structure.

3.10.1.21 Transition Dates

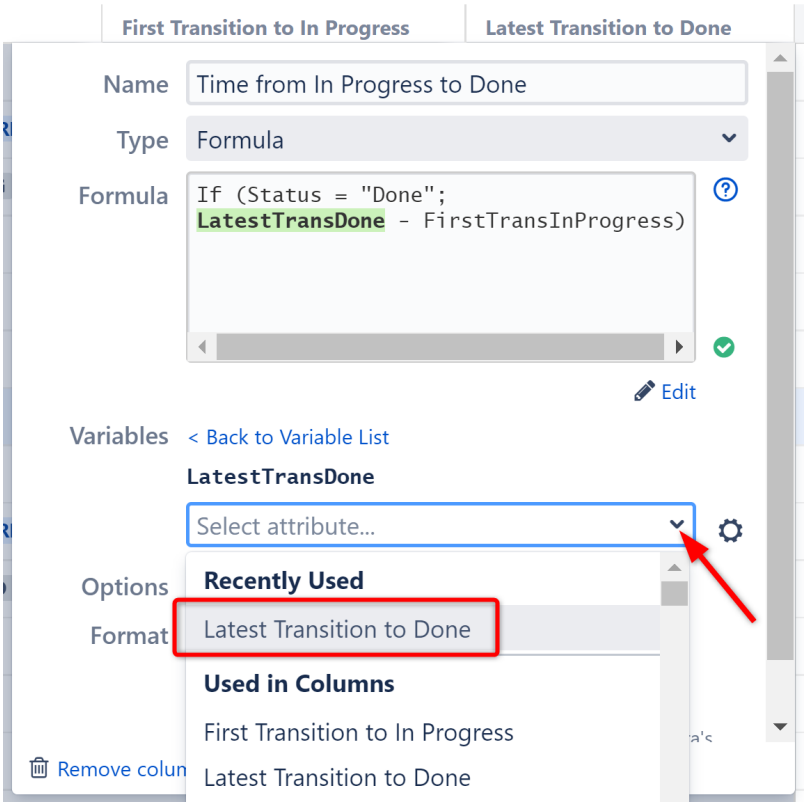
Transition Date Columns

The Transition Date (In Progress) and Transition Date (Done) columns allow you to visualize the dates issues were moved to In Progress or Done.



Transition Dates in Formulas

These Transition Date values can also be used in formulas by assigning a variable to the appropriate Transition Date attribute.



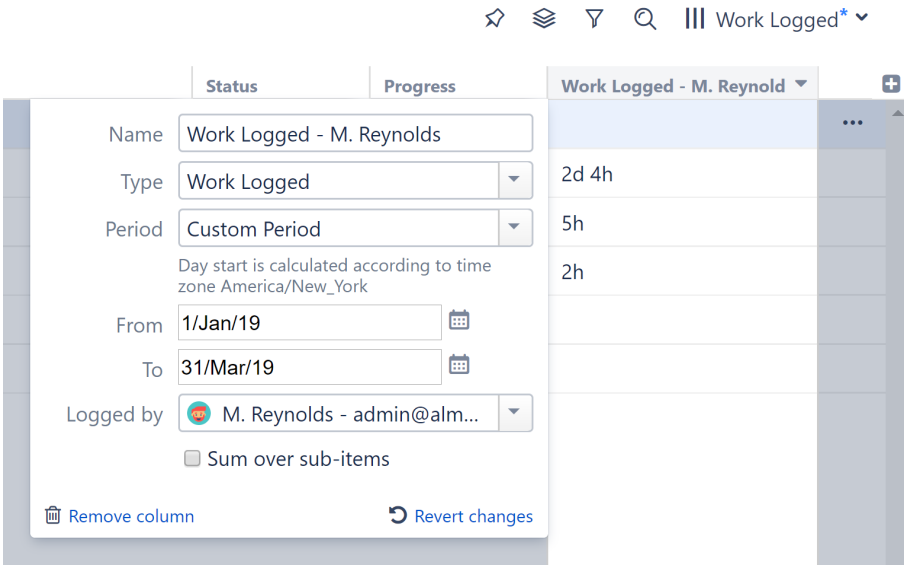
3.10.1.22 Work Logged Column

The Work Logged column displays the sum of time spent on an issue over a specific period of time and, optionally, by a specific user, group or project role.

Summary	Work Today
Epic 1	
Story 1	4h
Story 2	5h
Bug 1	2h
Story 3	1d

Customizing a Work Logged Column

Each Work Logged column can be customized to display precisely the information you need to see. Using the column configuration panel(see page 501), you can select from one of the predefined time periods or specify a custom period. Additionally, you can choose to narrow the results to a particular user, group or project role.



Displaying Aggregate Values

The Work Logged column can also display aggregated values, calculated as the sum of time spent over sub-issues.

To display aggregated values:

- From the Add Columns menu, select **Σ Worked Logged...**, or
- Starting from an existing Work Logged column, open the column configuration panel(see page 501) and select **Sum over sub-issues**

How is Work Logged Calculated?

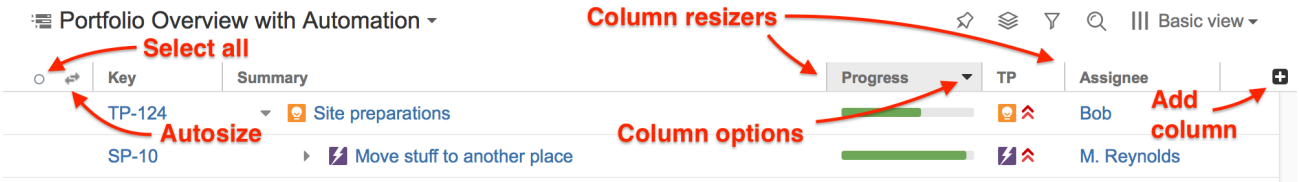
Each time you log work on an issue, you have to define "Time Spent" and "Date Started" values. The Work Logged column summarizes the logged time spent over a selected time period.

⚠ The start of the selected period is calculated based on the column creator's time zone. This time zone can be configured on the user's profile page[82].

- To view worked logged based on your own time zone, create your own instance of the Work Logged column.

3.10.2 Customizing Columns

To configure structure columns, position the mouse pointer over the structure header. The grid controls should become visible, allowing you to select and resize columns.

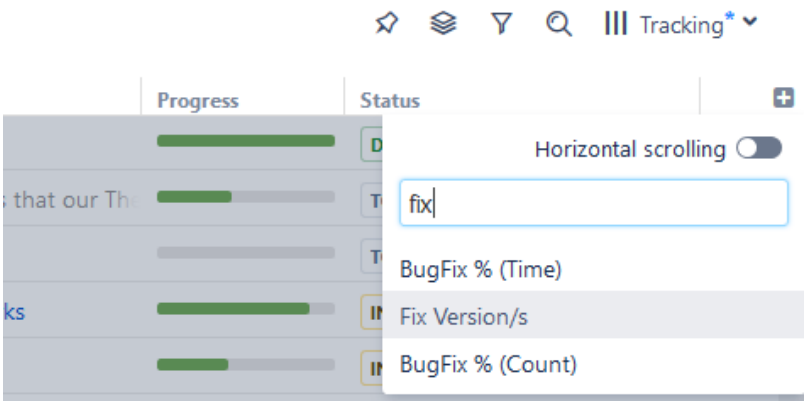


ⓘ When you add, configure, remove or rearrange columns, you make adjustments to the *View* that's being used to display the structure. An adjusted view is marked with a blue asterisk (*). The adjustments are stored in your browser and affect only yourself – to make the changes permanent and available to others, you need to save the view or create a new view(see page 511).

3.10.2.1 Adding Columns

To add a column, click on the **+** button at the right corner of the table header. A drop-down with the available column presets appears. To select the desired column, you can:

- Use the mouse to find a specific column,
- Use the keyboard **arrow keys** to select the column and hit **Enter** when done, or
- Start typing the column name to filter the list, and then select the appropriate column.



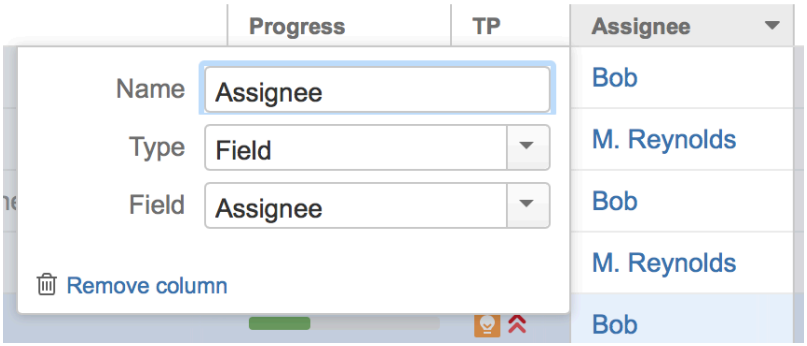
To abort adding a new column, hit **Escape**.

82 https://confluence.atlassian.com/display/JIRA/Managing+your+User+Profile

Use keyboard shortcut **TT** to quickly open the Add Column dialog (hit "t" twice).

3.10.2.2 Configuring Columns

To configure a column, click the arrow icon in the column header. The column will be highlighted, and its configuration drop-down will appear, allowing you to change the column name, type and other options.



The particular set of options available for the column is determined by its type(see page 458). For example, the Field(see page 461) column type lets you select the issue field to display and enable aggregation for numeric and time-tracking fields.

Any changes you make are applied immediately, so you can see the effect almost instantly. When you are happy with the column, simply close the configuration panel by clicking the arrow icon again or clicking anywhere outside the panel.

To cancel all of your changes use the **"Revert changes"** button at the bottom of the configuration panel. The column will be restored to its original state.

3.10.2.3 Removing Columns

To remove a column, click the arrow icon in the column header, then use the **"Remove"** button at the bottom of the column configuration panel.

You cannot remove the Summary Column(see page 490), Flags Column(see page 461) or Jira Actions Column(see page 469).

3.10.2.4 Rearranging Columns

You can change the position of a column by grabbing the column name with the mouse and dragging it to the left or right.

3.10.2.5 Resizing Columns and Autosize

Structure automatically tries to give all displayed columns enough space to display all information, but sometimes you might need to give more space to a column or two.

There are a number of ways to change column widths:

- **Grab the resizer and drag.** When you hover your mouse over a column, the resizer that is responsible for that column's width is highlighted. When the column size is close to what Structure considers ideal width (based on the displayed data), the resizer "snaps" to the perfect position.
- **Hold CTRL and drag resizer.** Works the same as above, but without snapping. You can use it to fine-tune column width.
- **Hold ALT (Option) and drag resizer.** In this mode, you will redistribute the space between two adjacent columns - increasing the width of one column and decreasing the width of another.
- **Double-click the resizer or the column header.** The column will automatically resize to the default size.
- **Click the "Autosize" icon (⇄) or double-click the Summary column.** All columns will be resized automatically, based on the displayed data.

If you are unable to comfortably view all of the columns or need to stretch columns beyond the visible panel, you may need to enable Horizontal Scrolling(see page 503).

⚠ Column widths are not part of the *View* and are not saved on the server or shared.

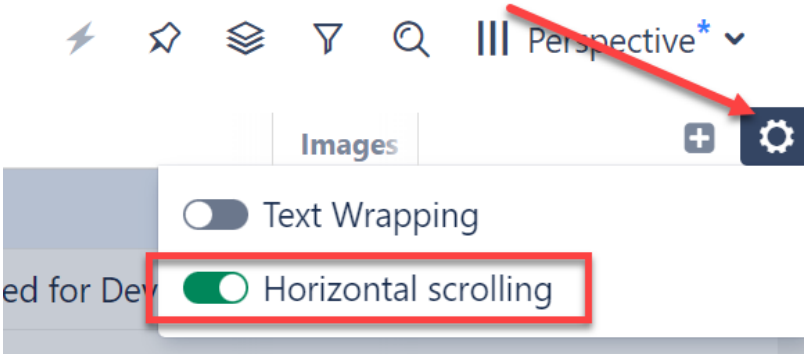
3.10.3 Horizontal Scrolling

Structure automatically adjusts column sizes to fit every selected column into the panel. While you can easily change the size of each column, by default you are still bound by the size of your Structure panel. This can make it difficult to view several columns at once or columns with a great deal of information (such as the Summary column or text columns), particularly when you're working with a Double Grid layout or viewing Structure on a small screen.

To make it easier to work with multiple columns and/or larger columns, you can turn on horizontal scrolling. This will allow you to work with as many columns as you need and set their sizes as large as necessary for easy viewing.

To enable horizontal scrolling:

- Click the gear icon in the top right corner of the Structure grid.
- Click the **Horizontal scrolling** toggle.



Once you turn horizontal scrolling on, your view may no longer fit within the viewing panel. In this case, a horizontal scrollbar will appear at the bottom of the panel. Sliding this from left to right will allow you to bring different columns into view.

3.10.3.1 Pinned Columns

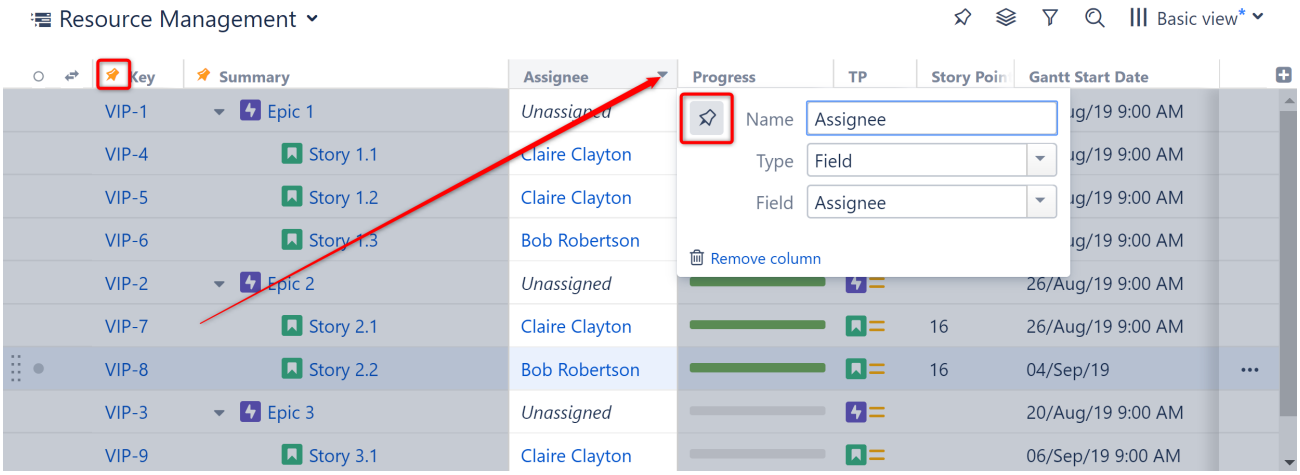
Not all columns move when you scroll. The Summary column(see page 490) remains visible at all times, so you can keep track of essential information as you focus in on other columns.

Additionally, the Jira Actions column(see page 469) is always available, though it may not be visible.

Pinning Additional Columns

To pin additional columns:

- Open the column's configuration menu and click the Pin icon. The column will be added to the pinned section in the farthest-right position.
- Drag the column to the left of any pinned column. The column will be added to the location you drop it.



Pinned columns will be moved to the left side of the screen and will remain visible as you scroll.

Unpinning Columns

To unpin a column:

- Open its configuration menu again and unclick the Pin icon. The column will be moved back to it's pre-pinned position.
- Drag it to the right of an unpinned column. The column will be placed wherever you drop it.

Additionally, when Horizontal scrolling mode is turned off, all pinned columns will be moved back to their original positions, when possible.

> The Summary column cannot be unpinned.

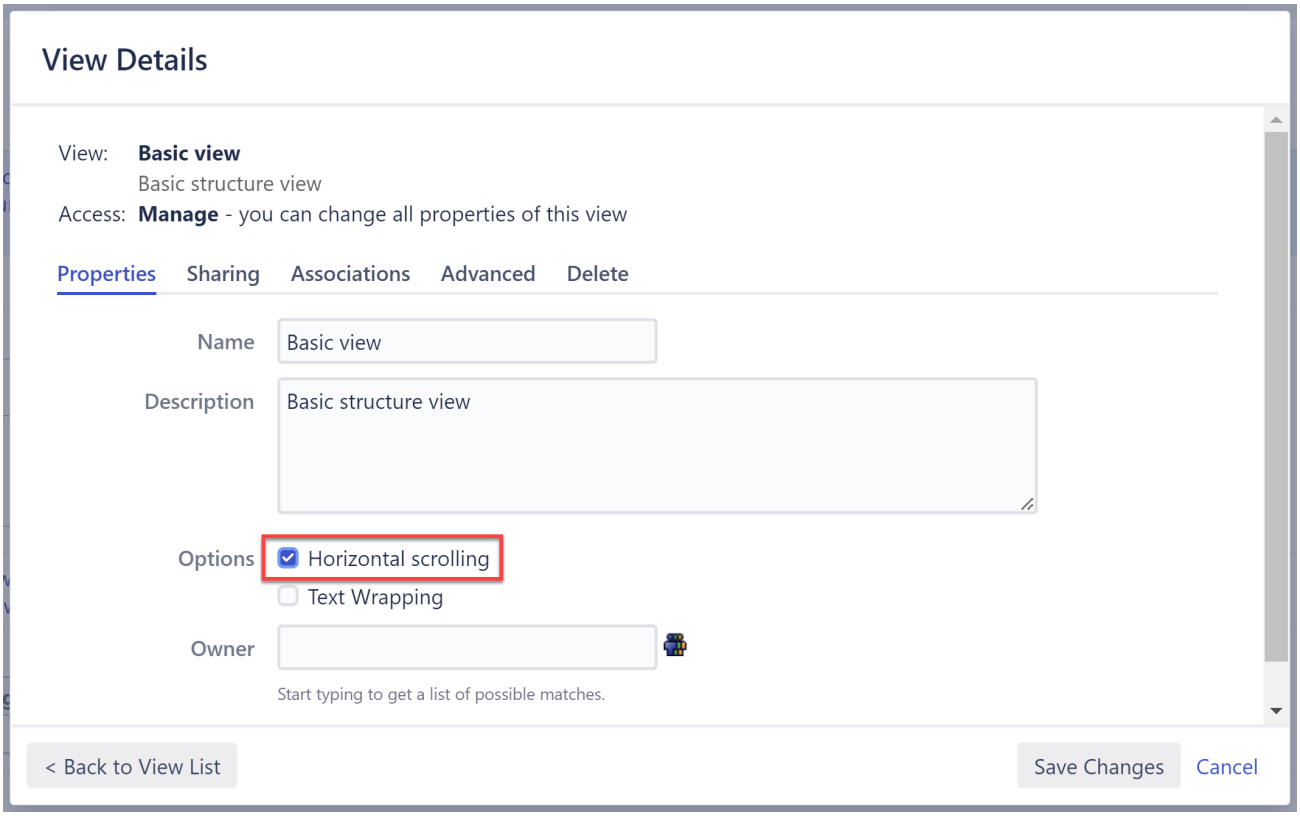
3.10.3.2 Saving Horizontal Scrolling Settings

Structure allows you to save your horizontal scrolling settings for a particular view.

For example, by default, the Basic View only shows a few essential columns. Keeping horizontal scrolling disabled for this view makes sure all of your essential information is visible at a glance. On the other hand, the Planning view often has several columns and might be easier to scroll through than trying to view all together.

To set your horizontal scrolling preference for a saved view:

- In the Views Menu, select **Manage Views**
- Locate the view you want to set and click **Details**
- Under the Properties tab, check or uncheck **Horizontal scrolling**

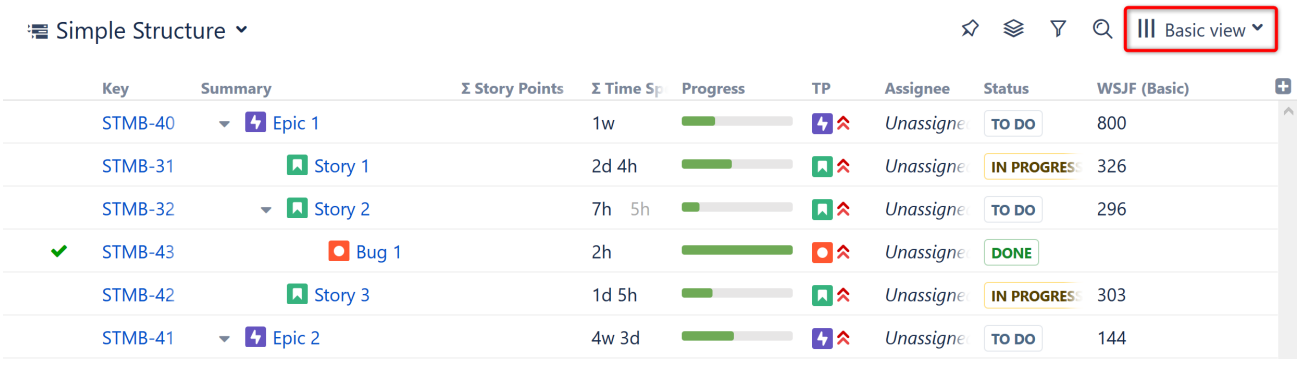


Horizontal Scrolling is enabled by default for the Planning and Tracking views. If you prefer not to use Horizontal Scrolling for these views, you can disable it using the methods above.

3.10.4 Managing Views

In Structure, a **view** defines which columns are displayed and in what configuration.

On the Structure Board, the current view is displayed in the top right corner. Click here to select a new view or save changes to the current view.

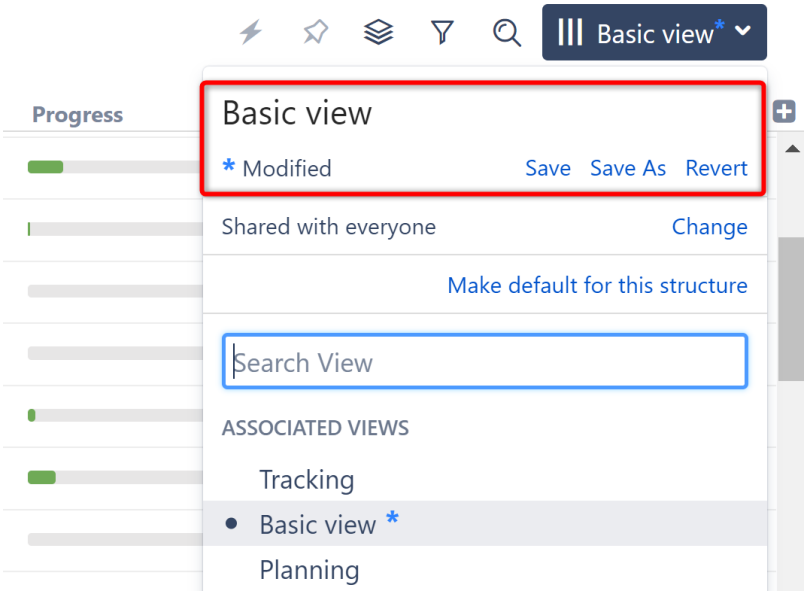


When viewing a structure from other locations in Jira, due to space limitations the view's name may not be displayed. To identify the current view, simply hover over the Views icon (the 3 vertical lines).

3.10.4.1 Changing the View

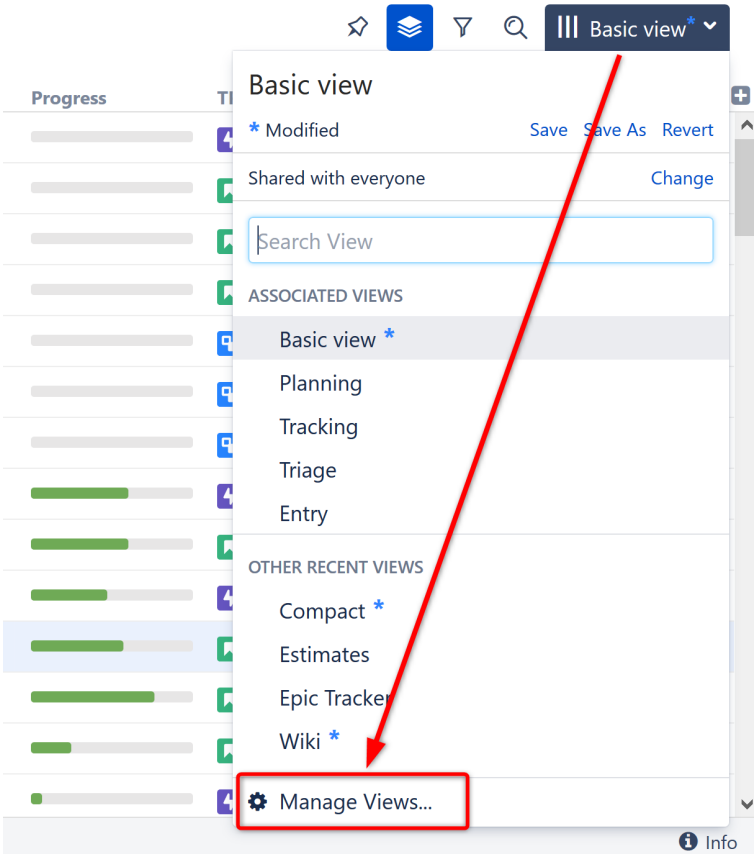
To change which columns are displayed or the order in which they appear, you can select a new view(see page 507) or manually add, remove or rearrange columns(see page 501).

When you manually change the column configuration, you create local adjustments to the currently-selected view. You can then save the changes (if you have permissions to change the view) or save and share your customization as a new view – see Saving and Sharing Views(see page 511).



3.10.4.2 Opening and Managing Saved Views

You can find, select and save views from the Views menu. For additional operations, or to browse all available views, open the Views menu and select **Manage Views**:

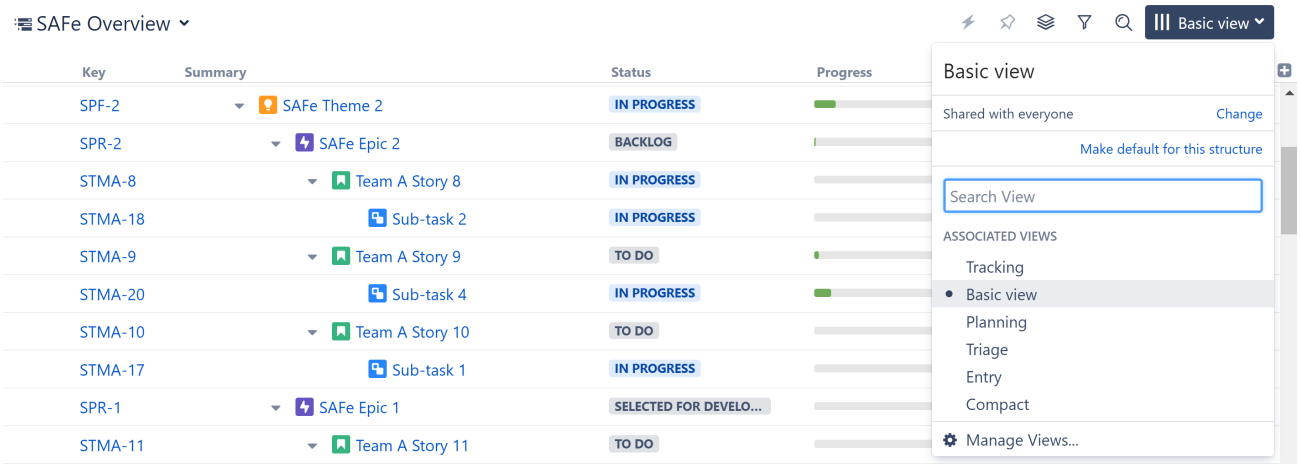


The following sections cover specific elements of view management:

- Views Menu(see page 507)
- Locating a View(see page 510)
- Saving and Sharing Views(see page 511)
- View Sharing and Permissions(see page 513)
- Changing View Settings(see page 515)
- Copying a View(see page 516)
- Deleting a View(see page 516)
- Associating Views with Structures(see page 516)

3.10.4.3 Views Menu

The Views menu allows you to change the current view, save changes to a view, share a view, and manage existing views.



To open the Views menu, click the Views icon (the 3 vertical lines in the top-right corner of the Structure panel). Depending on where you're viewing the structure, the current view's name may be listed next to the Views icon.

In the Views menu, you will see the following:

1. The current view's name. Hover the mouse over the name to see the view's description.
2. If the view was modified, you'll see a corresponding message and links to save or revert these changes(see page 511). Additionally, in the Associated Views/Recent Views sections, a blue asterisk will appear next to the name.
3. Permissions settings and a link to change them.
4. The option to make the current view the default view (see page 509)for the structure, or an indicator that it is already the default view.
5. Search View. Search looks for any views that match the entered name, not only those in this list.
6. Associated Views list. This list can be customized for each structure by the structure owner or anyone who has Control access level (see page 543)for the structure – see Customizing View Settings(see page 541).
7. List of views you have recently used (excluding the views shown in the section above).
8. Manage Views link, which opens the View Management(see page 507) dialog.

Switching Views

To switch the current view, open the Views menu. From there, you have three ways to choose the new view:

1. Select it from the drop-down list. (This is the easiest method, but it may not include all available views.)
2. Search for it in the Search View panel.
3. Click **Manage Views...** to see all available views.

Switching View with Keyboard

You can also switch the current view using only the keyboard:

1. Use the **vv** shortcut to open the Views menu (hit "v" twice).
2. Use the arrow keys to select a view, or enter text to search for matching views.
3. Hit **Enter** to switch to a selected view or **Escape** to close the menu.

To go back to the previous view, use the **vvv** shortcut (hit "v" three times).

Opening a Structure

When a structure opens, it will select a view based on the history and settings for the location the view is being opened (Structure Board, Issue page, etc.). Structure will attempt to assign a view in the following order:

1. The last view you used for that structure in that location, if available.
2. The last view used for that structure in the opposite panel (primary panel vs. secondary panel(see page 519)) in that location, if available.
3. The default view (see page 509)for the structure, if available and viewing the structure on the Structure Board.
4. The global default view, as set by the Structure administrator.

The same structure opened on an Issue page will typically open with a different view then if it were opened on the Structure Board, because users need to see different information in different locations.

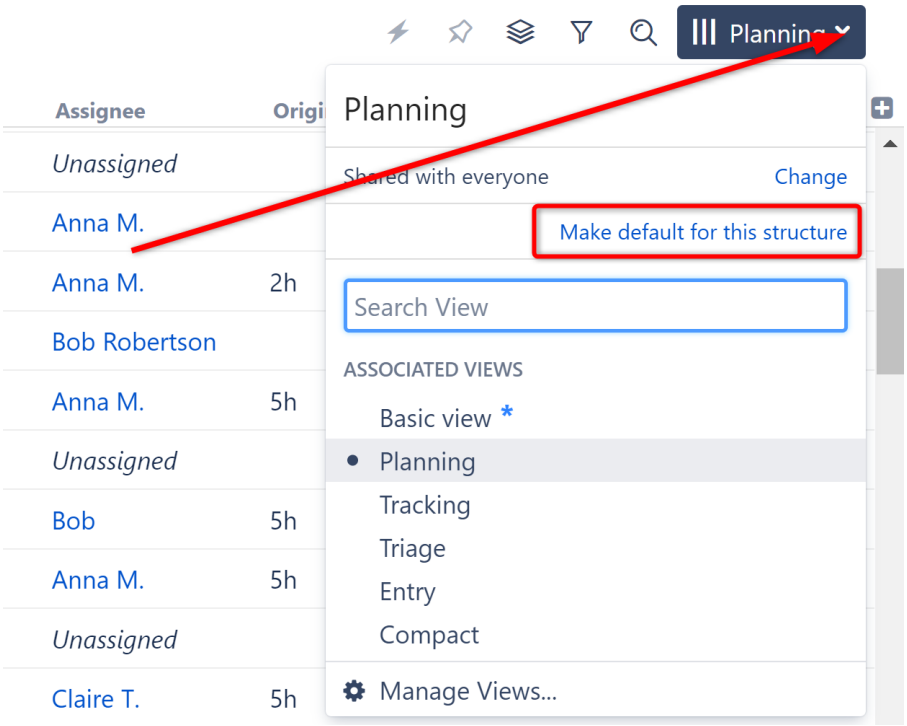
Default View

Users with Control permission(see page 543) can set a default view for the structure. When other users open a structure for the first time, they will see the default view. Additionally, if there is no recently-selected view available for the structure, it will open with the default view.

Setting the Default View

You can set the default view for a structure using Manage Structure (see page 541)or the Views menu. To set a default view using the Views menu:

1. Select the view you want to make the default. See Switching Views(see page 508) above.
2. Open the Views Menu again.
3. Click **Make default for this structure**.

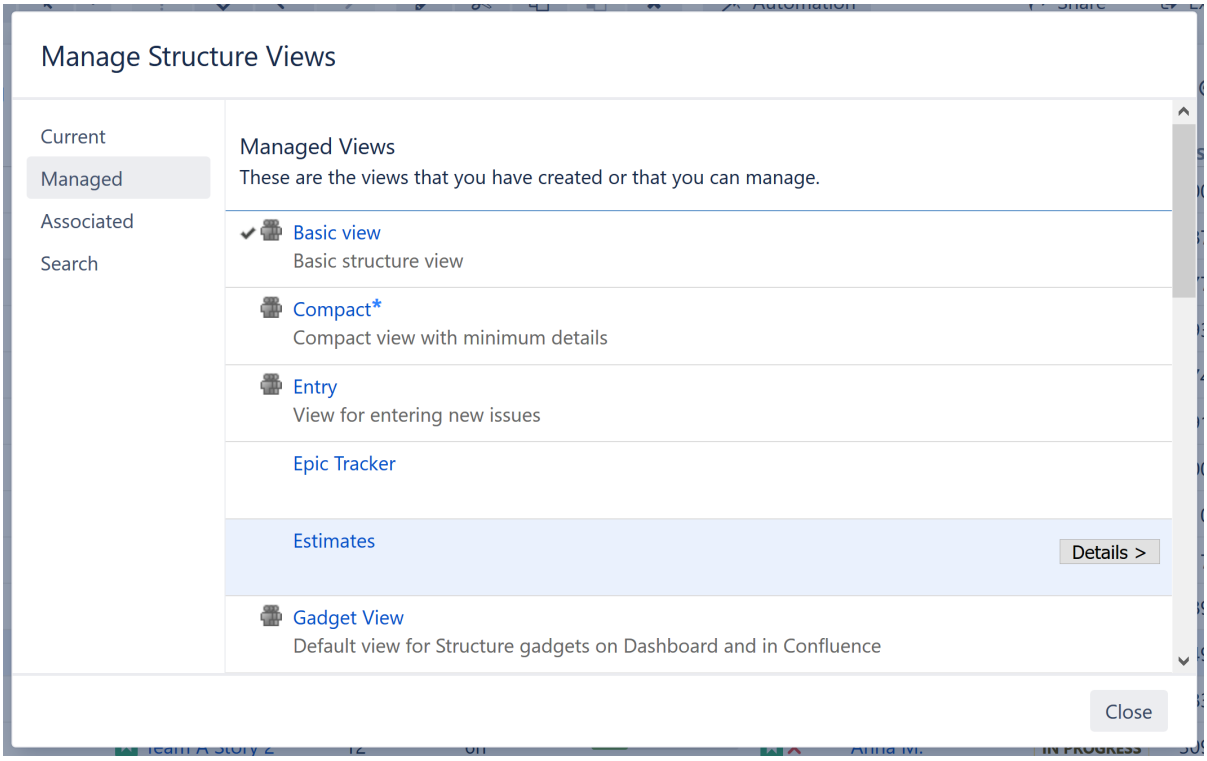


If the selected view is already default for the structure, this section will let you know that you are either "Using the default view" or "Using the default view with modifications."

If you want to return to the previously selected view, use the keyboard shortcut: **vvv** (press "v" three times).

3.10.4.4 Locating a View

The easiest way to locate a view is to find it in the drop-down list or search bar on the Views menu(see page 507); however, if you can't locate a particular view that way, or you need to make changes to a view, you can also locate views using the Manage Views dialog(see page 510).

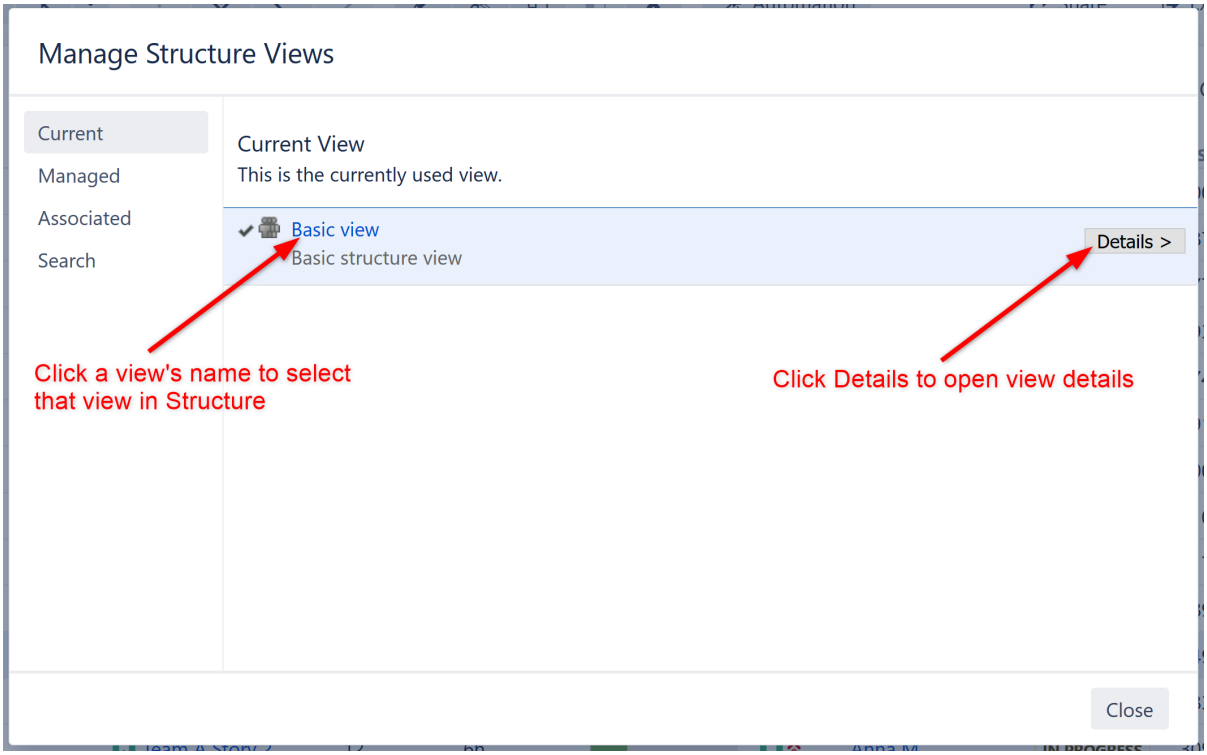


To find a particular view, select one of the following tabs:

- **Current** - displays only the view that is currently selected in the Views menu. You can quickly go to the current view's details from this tab.
- **Managed** - displays all views that you can **Manage** – that is, you have full administrative permissions(see page 513) for those views.
- **Associated** - displays all views that are associated with the currently-viewed structure (by the structure administrator).
- **Search** - allows you to search for views or display a list of all available views.

When you have located a specific view, you can click its name to switch to that view. The view will also appear in the **Other Recent Views** section of the Views menu(see page 507).

To see and edit View details, click the **Details** button that appears when you move the mouse pointer over the view record.



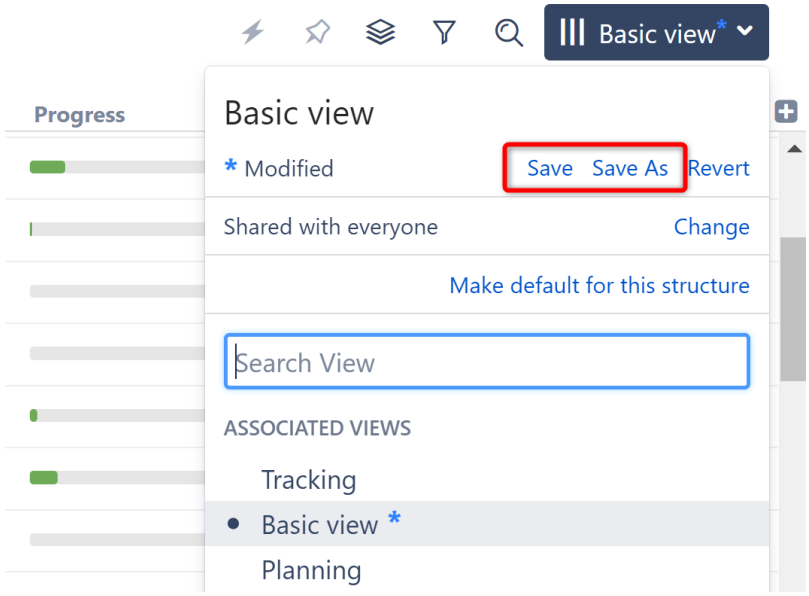
3.10.4.5 Saving and Sharing Views

When you add, remove or rearrange columns(see page 501), you are making changes to the currently-selected view. The view will now be marked with a blue asterisk, denoting that it has been modified from its original version.

These changes are stored locally and only visible to you. Anyone else using that same view will continue to see it in its original state, without your changes.

Saving View Adjustments

To make changes permanent and push them to other people using the same view, you need to save a new version of the view. To do so, open the Views menu and click the **Save** link. You can also select **Save As** to create a new view with the current changes, without affecting the original view.

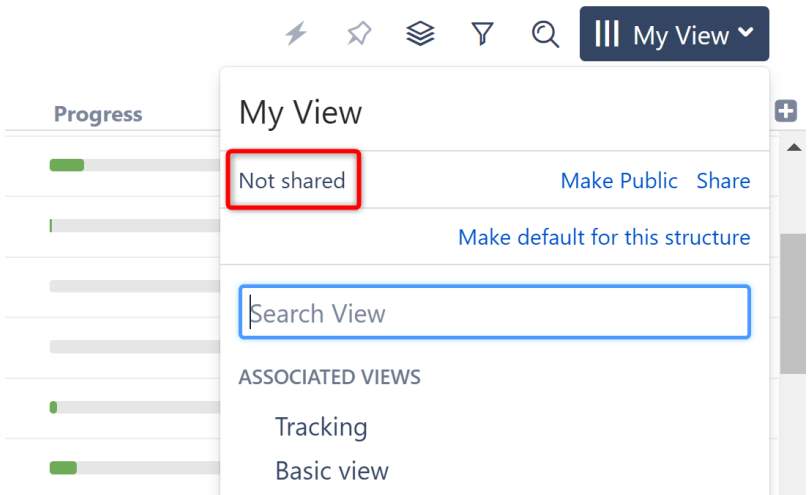


To save changes to an existing view, you need to have *Update* access level for that view (see View Sharing and Permissions(see page 513)). If you do not have permission to change the view, you can still create a new view based on your modifications with the **Save As** link.

If you need to remove your adjustments and return to the original view as it is stored on the server, click the **Revert** link.

Sharing a View

A view has a set of permissions, just like a structure. When you initially create a view with the **Save As** link, the view is **private**. You can use the view with any structure, but no one else can use the view.



To share a view with other people, open the Views menu and:

- Click the **Make Public** link to make the view **public**, allowing everyone to locate and use this view.
- Click the **Share** link to specify exactly which users can use, update and manage the view. See View Sharing and Permissions(see page 513) for details.

3.10.4.6 View Sharing and Permissions

Like structures, views can be shared with different levels of access for each group of users.

There are four levels of access to a view:

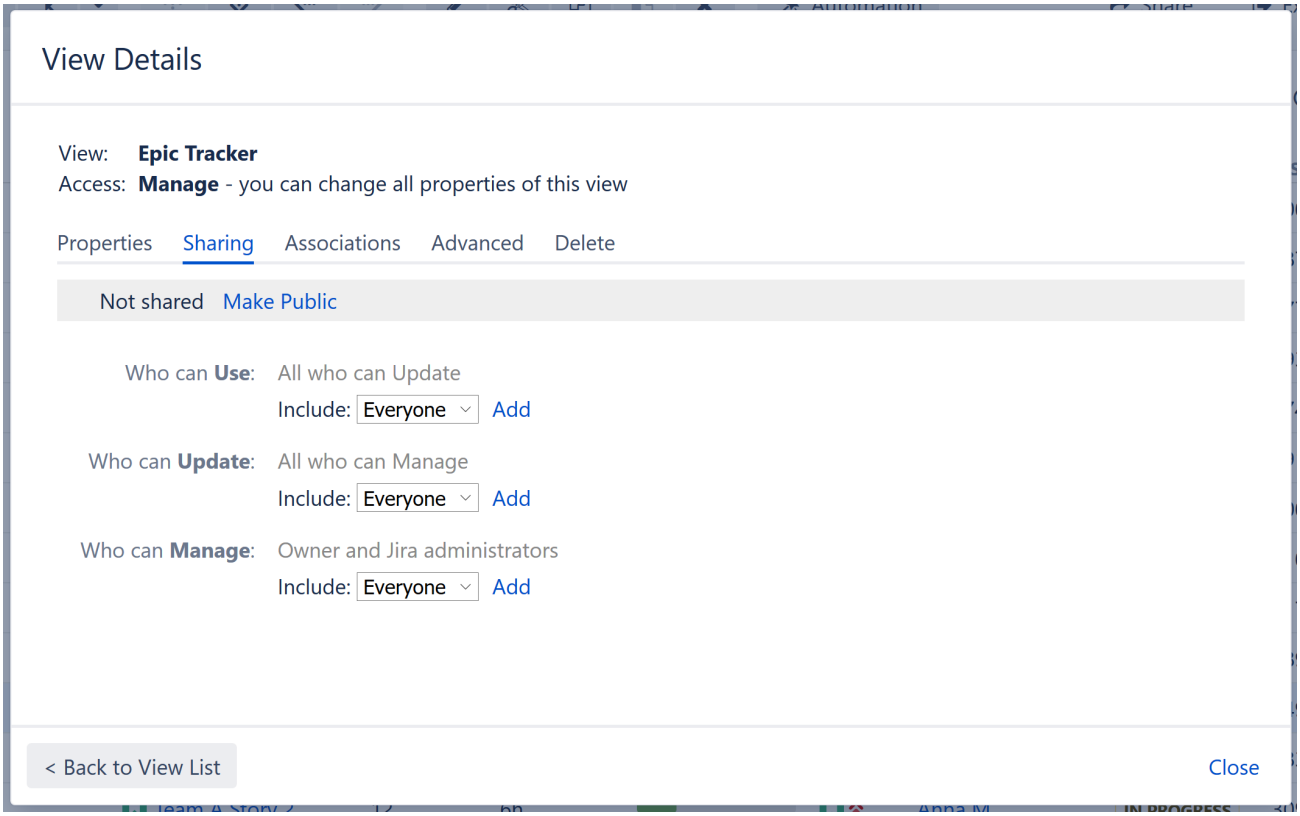
None	The view is not visible nor usable by the user.
Use	Read-only access: the user can use the view, but cannot modify it.
Update	The user can use the view, and also save view adjustments as the new version of the view. The user cannot modify view name or sharing permissions.
Manage	The user can change any of the view's properties and also can delete it.

⚠ View owner and Jira administrators always have **Manage** access to a view.

ⓘ People who have only **Use** permission for a view still **can** add, remove or rearrange columns(see page 501), but they can't save the modified configuration as a new version of the view. They can use the **Save As** link to create a new view with the modified configuration.

Changing permissions

If you have **Manage** access to a view, you can modify its permissions on the **Sharing** tab of the view details dialog.



For each level of access, you can define categories of users who have this type of access:

- Nobody
- Specific user groups
- Specific roles in specific projects
- Everyone (including anonymous users)

Higher-level access implies all lower-level access. So everyone who can **Manage** a view, can also **Update** and **Use** it - there is no need to add those users at all three levels!

Private and Public Views

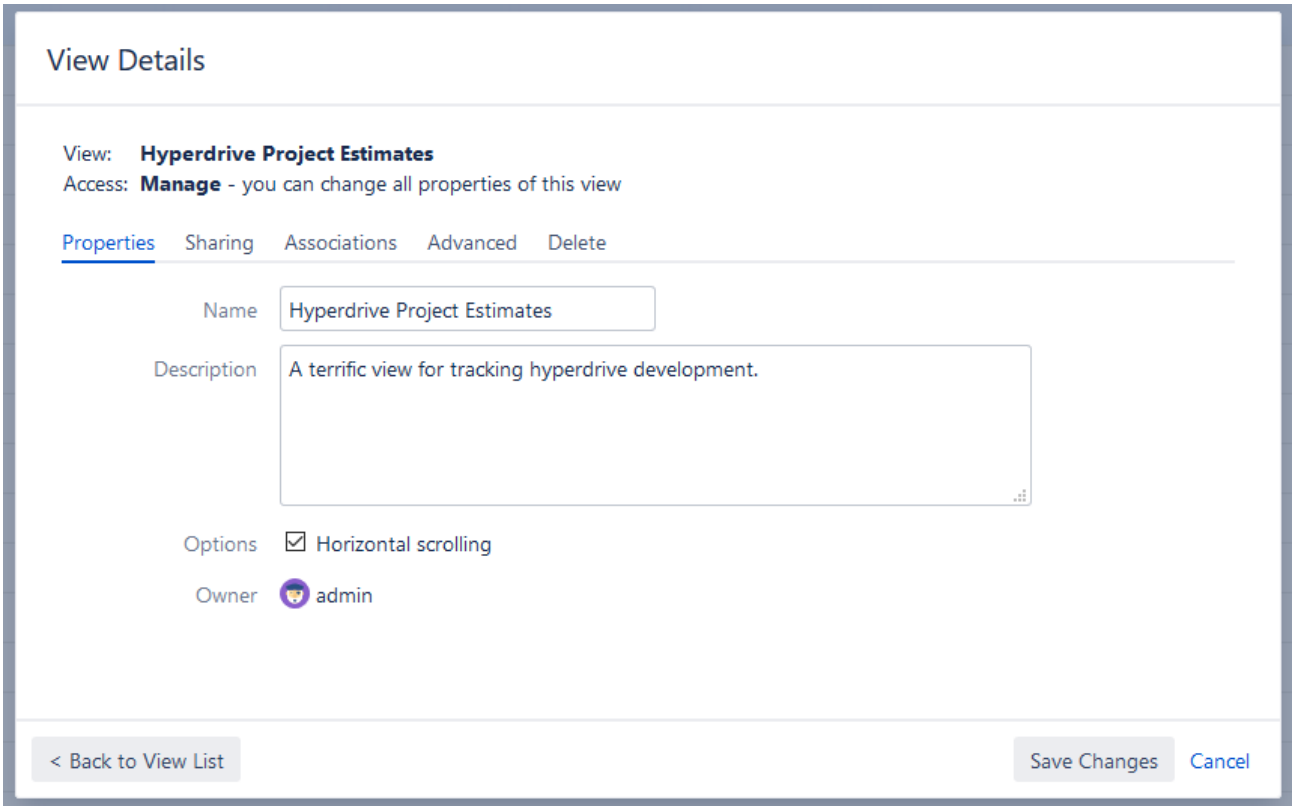
When a view is not shared with anyone, it's called a **private view**. You can quickly make a view private by clicking the **Make Private** link – this will remove all permission assignments.

When **everyone** is given at least **Use** permission for a view, it is called a **public view**. You can quickly make a view public by clicking the **Make Public** link on the the **Sharing** tab and also in the Views menu(see page 507) – this will give **Use** permission for that view to everyone.

You need to have global **Create Shared Objects** permission to be able to share views.

3.10.4.7 Changing View Settings

When you have located a view(see page 510) in the Manage Views dialog, click the **Details** button to open the View Details page in the same dialog:



The View Details page shows a number of tabs:

- **Properties** - lets you change the name and the description of the view, as well as select whether Horizontal Scrolling(see page 503) is enabled by default.
- **Sharing** - lets you view and modify sharing permissions for the view – see View Sharing and Permissions(see page 513).
- **Associations** - shows the structures which are associated with the view (have this view in their Views drop-down). See Associating Views with Structures(see page 516).
- **Advanced** - shows additional technical information about the view.
- **Delete** - lets you delete this view(see page 516).

> ⚠ The tabs and the scope of functionality available may be limited, depending on your access level to the view.

Renaming a View and Changing Other Properties

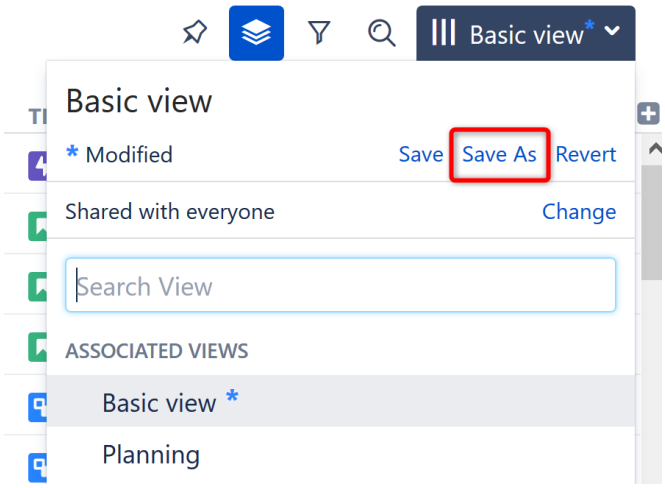
When you change a view's name, description, sharing permissions or anything on the Advanced tab, the changes are not saved until you click the **Save Changes** button. After you have saved the changes, they take effect for you and anyone else who has access to the view.

The Associations tab is different – it contains only links to structures. The associations between structures and views are managed by the structure administrator on the Manage Structure(see page 532) page.

3.10.4.8 Copying a View

There's currently no way to directly copy a view, but you can use the **Save As** function to create a new view based on the existing view configuration:

1. On the Structure Board, select a view you'd like to copy, so it is your current view. You can use the Views menu(see page 507) or Manage Views dialog(see page 510) to find the view you need.
2. If you don't have local adjustments to the view, make some – for example, add a column, or change column order. (Note that just resizing a column does not change a view configuration.)
3. Open the Views menu and use the Save As(see page 511) link to create a new view.
4. Use the **Manage Views** dialog to review the new view's description and sharing permissions.



3.10.4.9 Deleting a View

To delete a view:

1. Open the Manage Views(see page 516) dialog.
2. Locate the view(see page 510) you'd like to delete and click the **Details** button.
3. Open **Delete** tab and click **Delete This View**.

Deleting a view cannot be reverted.

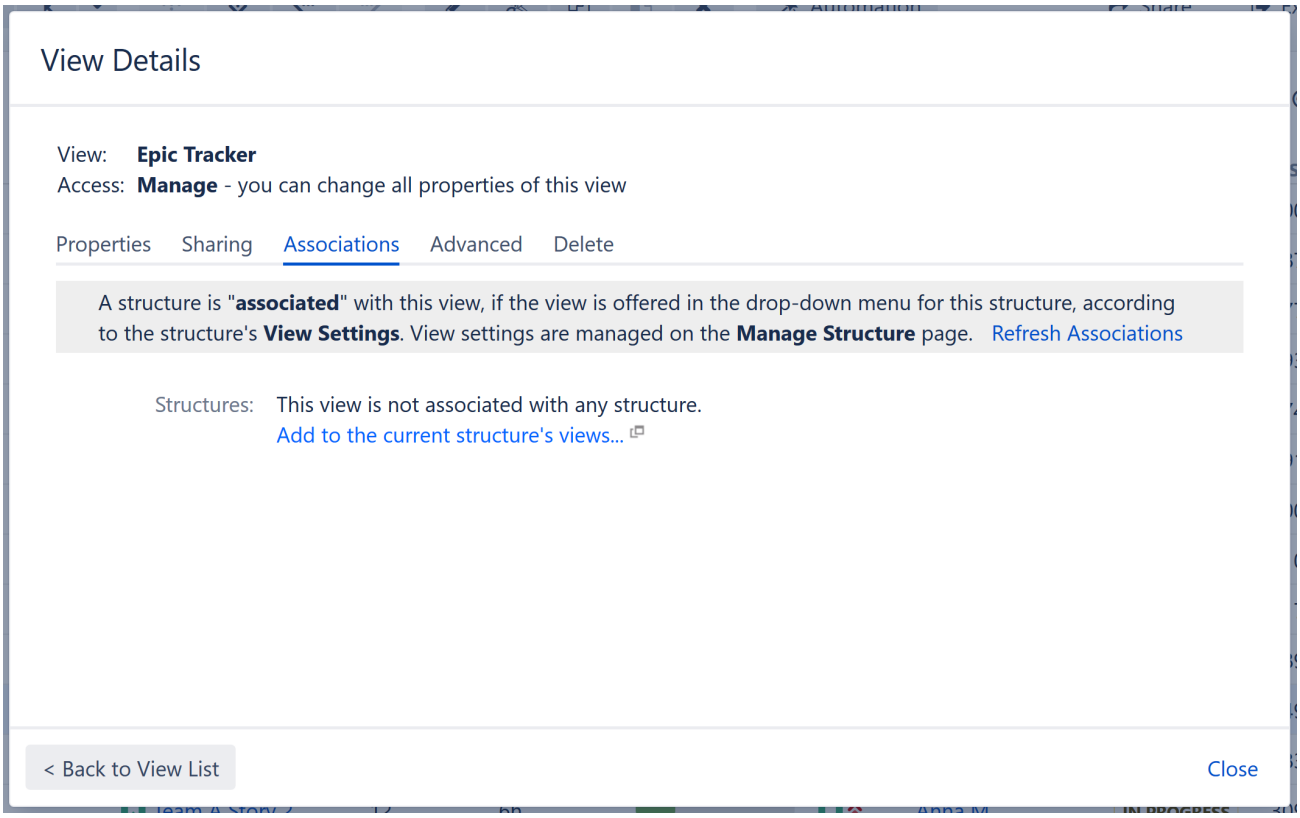
Before deleting a view, open the Associations tab(see page 516) and review the list of structures that are associated with this view. The associations won't prevent you from deleting the view, but you may want to discuss the matter with the administrators of those structures.

3.10.4.10 Associating Views with Structures

Users with **Control** access to a structure can associate particular views with that structure. These views will appear in the **Associated Views** section of the Views menu(see page 507) when that structure is used.

> Jira administrator can also specify global default view settings(see page 959), which define associated views for structures that don't have customized view settings.

To view which structures a view is associated with, or to associate a view with the current structure, open the Manage Views dialog(see page 516) and locate the view to associate. Click **View Details** and select the **Associations** tab.



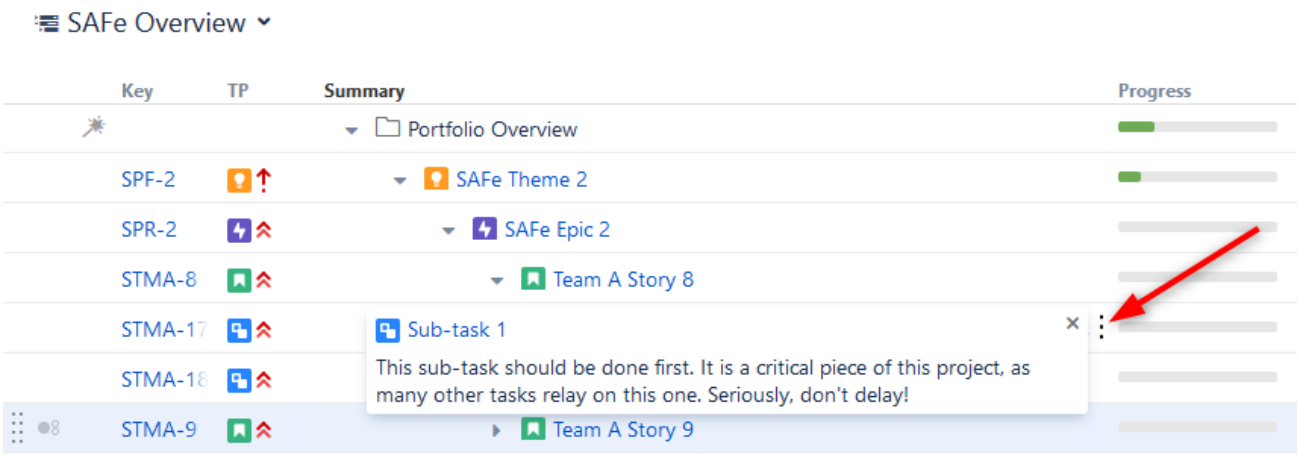
To associate the view with the current structure, click the **Add to the current structure's views...** link. If you have Manage access for the structure, this will open the View Settings(see page 541) for the structure and add the current view. Scroll to the bottom of the page and click **Apply**.

> View settings (associations between a view and a structure) are a property of the structure, not the view. The **Associations** tab on the View Details dialog is provided for convenience.

3.10.5 Displaying Full Cell Content

In the Structure grid, if the content of a cell is larger than the cell's size, only a part of the content will be shown.

You can view the full content by clicking or hovering the mouse pointer over the "More" sign (three vertical dots) that appears at the right side of the cell.



To close the full-content panel, click the x, move the mouse away, press Esc or click anywhere outside the panel.

You can start editing the cell value even when the full-content panel is shown: double-click the panel or the Summary text in the panel.

3.10.6 Double Grid Mode

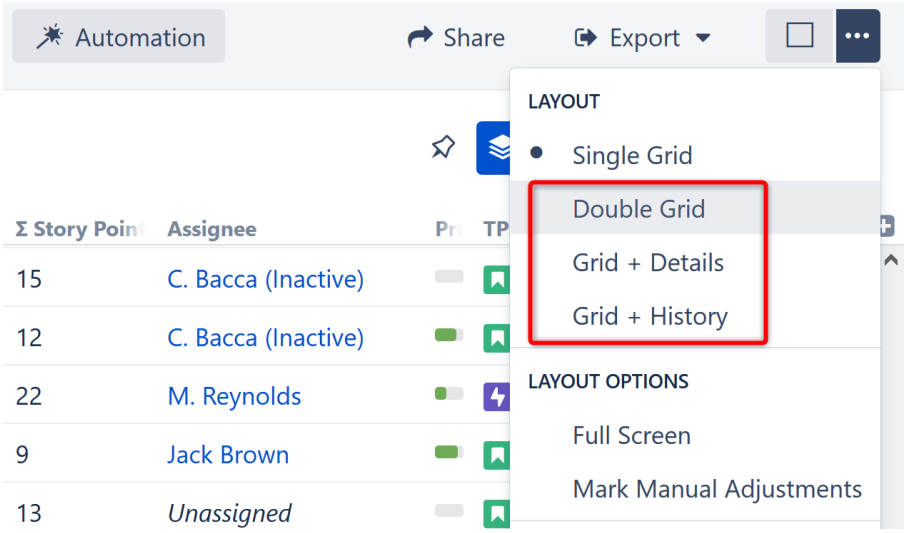
When working with a structure on the Structure Board, you can switch to the double grid mode to take full advantage of the screen space.

While the left panel always displays the structure widget or search, the right panel can open any of the following:

- Another structure, so you can work with two structures side by side
- A text/JQL search
- Clipboard contents
- Issue details(see page 126) panel - as you click an issue in the structure, you can see/edit the issue details in the panel on the right
- History(see page 556) - see the list of changes made to the structure and navigate through them to see the previous versions of the structure
- Add-on Information:
  - With Structure.Gantt installed, you can view a Gantt chart in the secondary panel
  - With Structure.Pages installed, you can display Confluence page contents

3.10.6.1 Viewing the Secondary Panel

To open the Secondary panel, use the **Toggle Panels** menu in the Structure Toolbar(see page 100).



You can select from the following options:

- **Double Grid** opens the secondary panel with the structure widget. By default, the widget opens with the JQL search. You can switch to text search, clipboard or another structure by clicking the JQL label.
- **Grid + Details** opens the Issue Details Page(see page 126) for the currently-selected issue.
- **Grid + History** opens Structure History(see page 556).
- If you have additional add-ons that utilize the secondary panel, their options will be displayed below these.

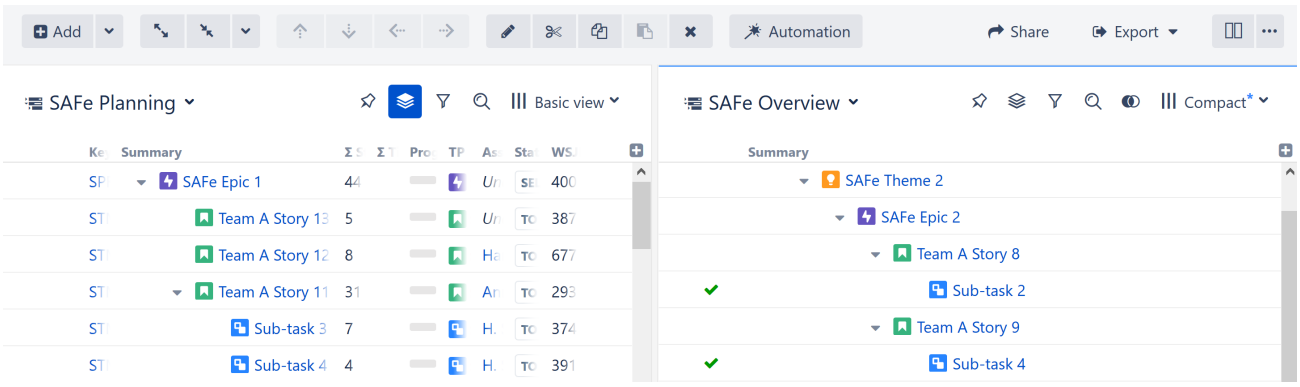
3.10.6.2 Resizing the Secondary Panel

You can divide the horizontal space between the secondary panel and the main panel by dragging the separating border.

3.10.6.3 Structure Widget on Secondary Panel

The structure widget that you open in the secondary panel is fully functional and differs very little from the widget in the main panel on the left. In both of them you can open structures, run JQL and Text search and open the clipboard.

Just like the main panel, it has its **panel toolbar** and the **Views menu**.



You can also use the Main Structure toolbar actions to work with the secondary panel widget. The toolbar actions will be applied to the panel that is in focus. The focused panel is highlighted with a thin blue line at the top.

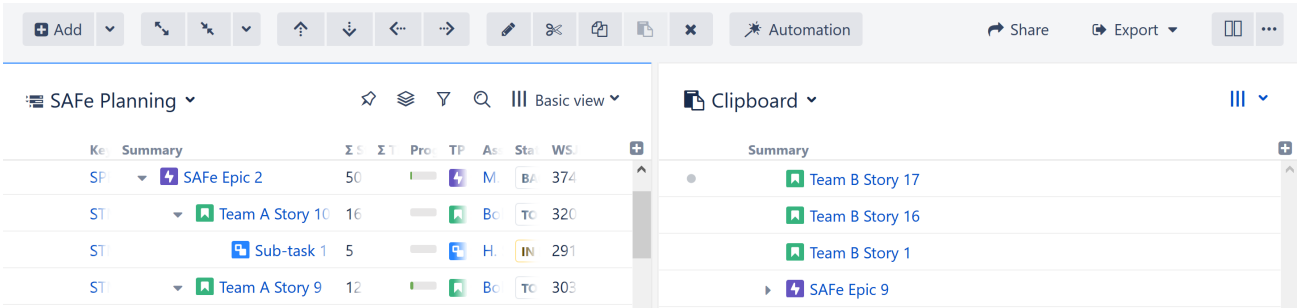
Hide/Show

The toolbar of the secondary panel has one extra function, Hide/Show. Clicking this button shows/hides items that are present in the main panel.

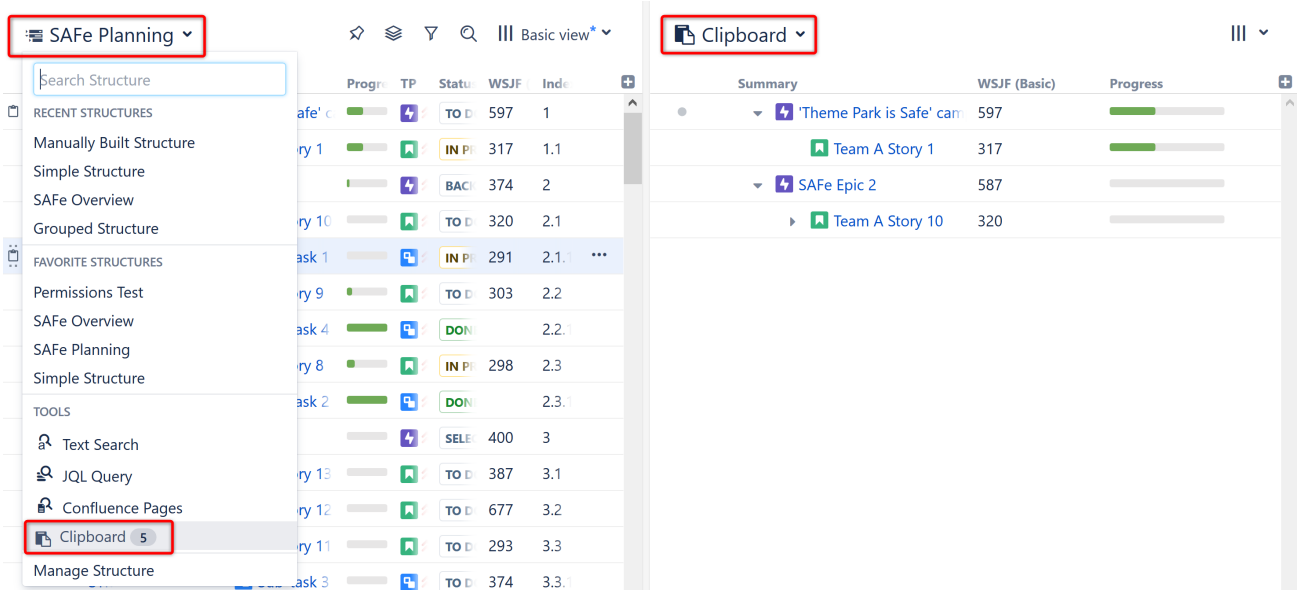
This is especially useful when you need to make sure the structure in the main panel has all the issues you've found using search in the secondary panel.

3.10.6.4 Issue Clipboard

The Issue Clipboard allows you to view issues that you have copied or cut. Clipboard contents can be viewed in the main Structure panel or the secondary panel.



To view the Issue Clipboard, click the structure name or the search-type label at the top of the panel and select **Clipboard** from the menu.



To learn more, see Cut, Copy and Paste(see page 110).

3.10.7 Two-Panel Mode

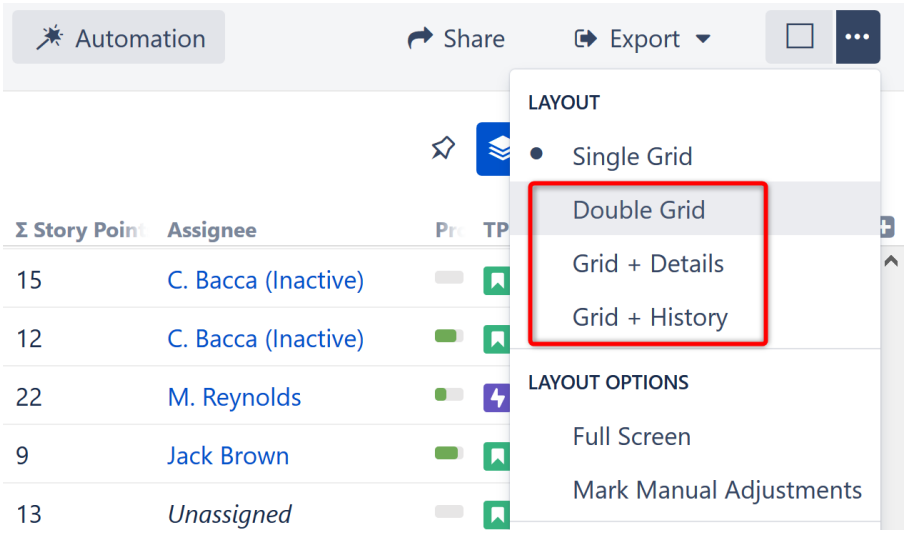
When working with a structure on the Structure Board, you can switch to the two-panel mode to take full advantage of the screen space.

While the left panel always displays the structure widget or search, the right panel can open any of the following:

- Another structure, so you can work with two structures side by side
- A text/JQL search
- Clipboard contents
- Issue details(see page 126) panel - as you click an issue in the structure, you can see/edit the issue details in the panel on the right
- History(see page 556) - see the list of changes done to the structure and navigate through them to see the previous versions of the structure
- Add-on Information:
    - With Structure.Gantt installed, you can view a Gantt chart in the secondary panel
    - With Structure.Pages installed, you can display Confluence page contents

3.10.7.1 Viewing the Secondary Panel

To open the Secondary panel, use the **Toggle Panels** menu in the Structure Toolbar(see page 100).



You can select from the following options:

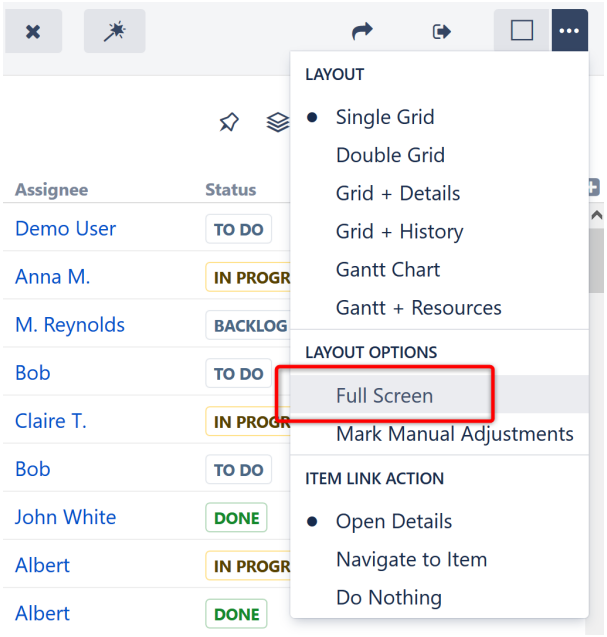
- **Double Grid** opens the secondary panel with the structure widget. By default, the widget opens with the JQL search. You can switch to text search, clipboard or another structure by clicking the JQL label.
- **Grid + Details** opens the Issue Details Page(see page 126) for the currently-selected issue.
- **Grid + History** opens Structure History(see page 556).
- If you have additional add-ons that utilize the secondary panel, their options will be displayed below these.

3.10.7.2 Resizing the Secondary Panel

You can divide the horizontal space between the secondary panel and the main panel by dragging the separating border.

3.10.8 Full Screen Mode

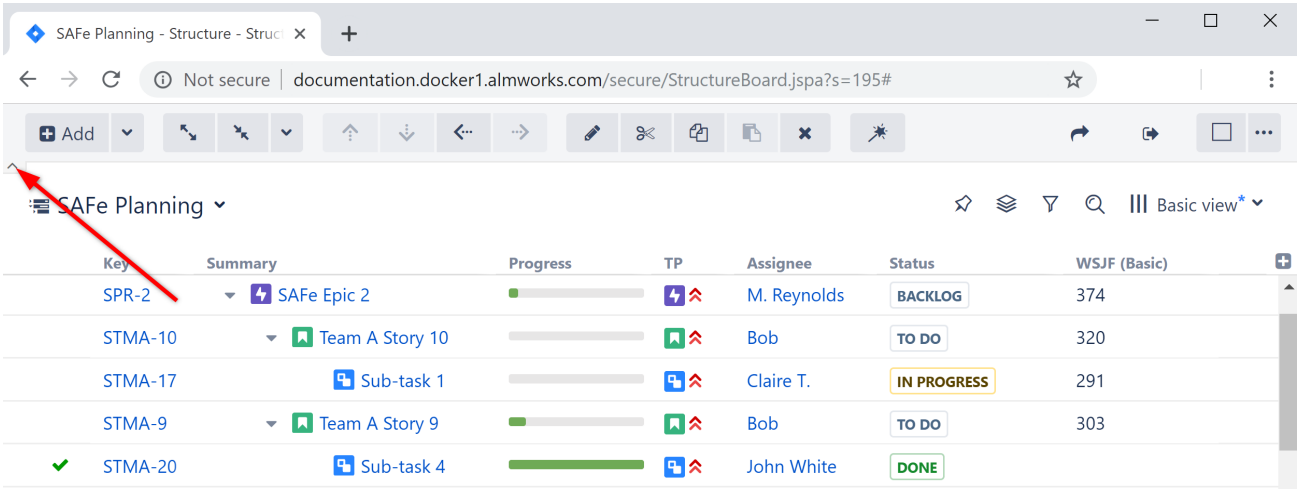
When working with the Structure Board you can turn on Full Screen mode to give more screen space to your data. Full Screen mode can be toggled using the **Toggle Panels** menu or by pressing Z on your keyboard.



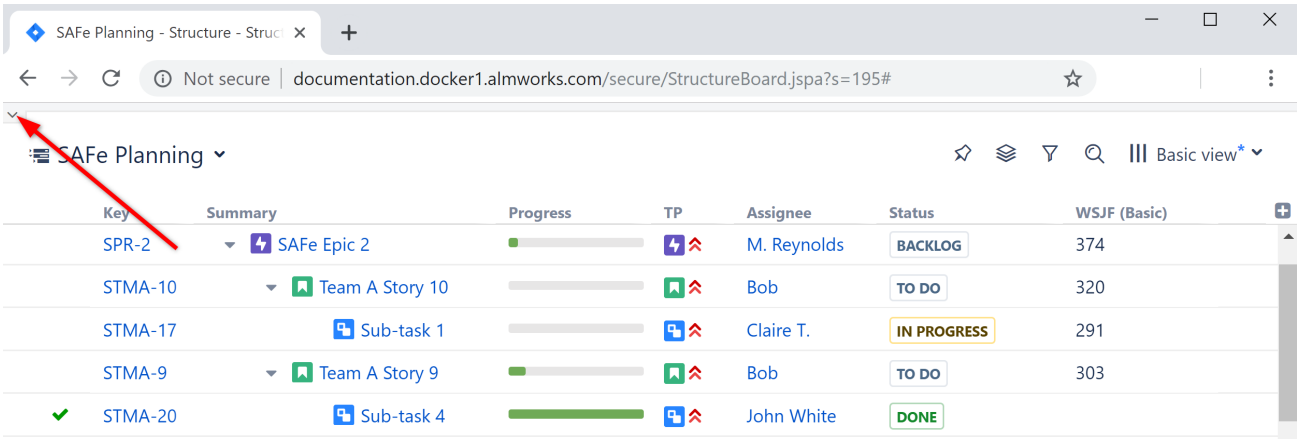
In Full Screen mode, the Jira application header is hidden and the Structure toolbar becomes more compact. To exit Full Screen mode, uncheck **Full Screen** in the Toggle Panels menu or press Z again.

3.10.8.1 Hide the Structure Toolbar

To save even more screen space, you can collapse the main toolbar by clicking the Collapse button.

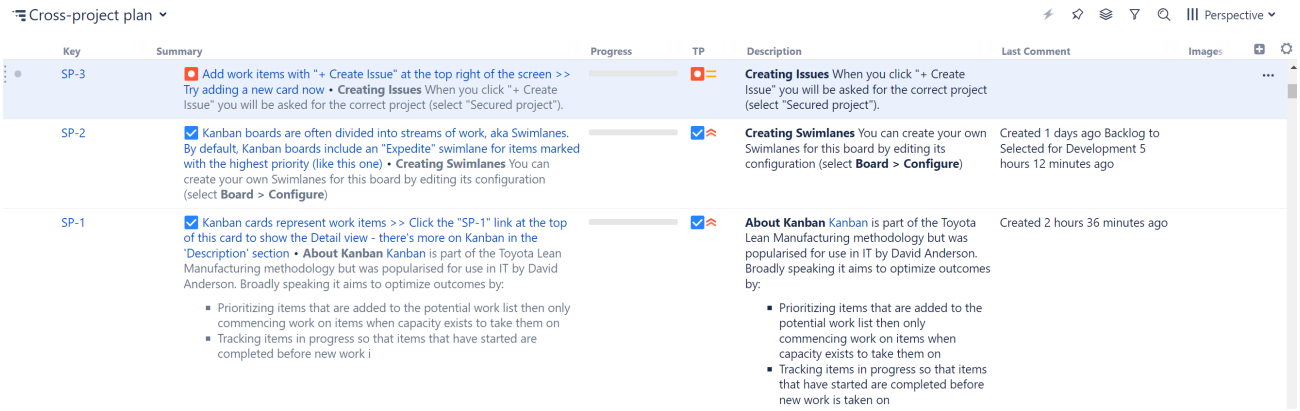


To use the toolbar in Collapsed mode, simply hover the mouse over the collapsed toolbar. It will reappear until you move the mouse away. To return the toolbar permanently, click the Expand button (where the Collapse button used to be).



3.10.9 Text Wrapping

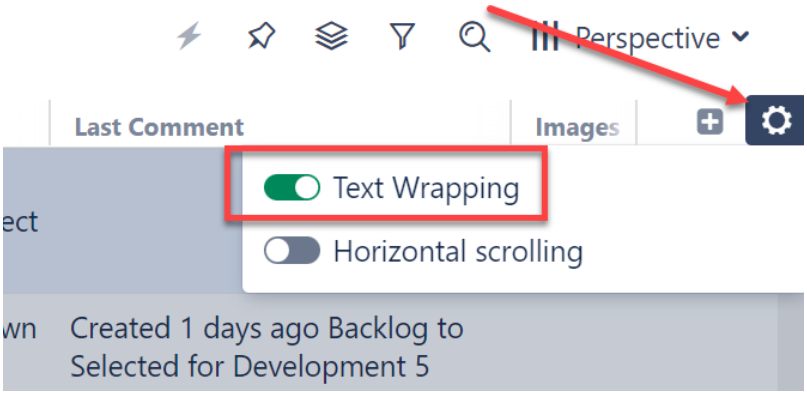
By default, the rows in a structure are only tall enough to display a single line of text for each column. If your columns contain long texts or large images, you can enable Text Wrapping to see more of each field. When enabled, each row's height is adjusted to fit its tallest column.



Text wrapping works with any text field, including Summary, Description, Last Comment, and custom text fields. It also works with formula columns and wiki markup(see page 251).

3.10.9.1 Enabling Text Wrapping

To enable or disable Text Wrapping, click the gear icon in the top right corner of the Structure grid.

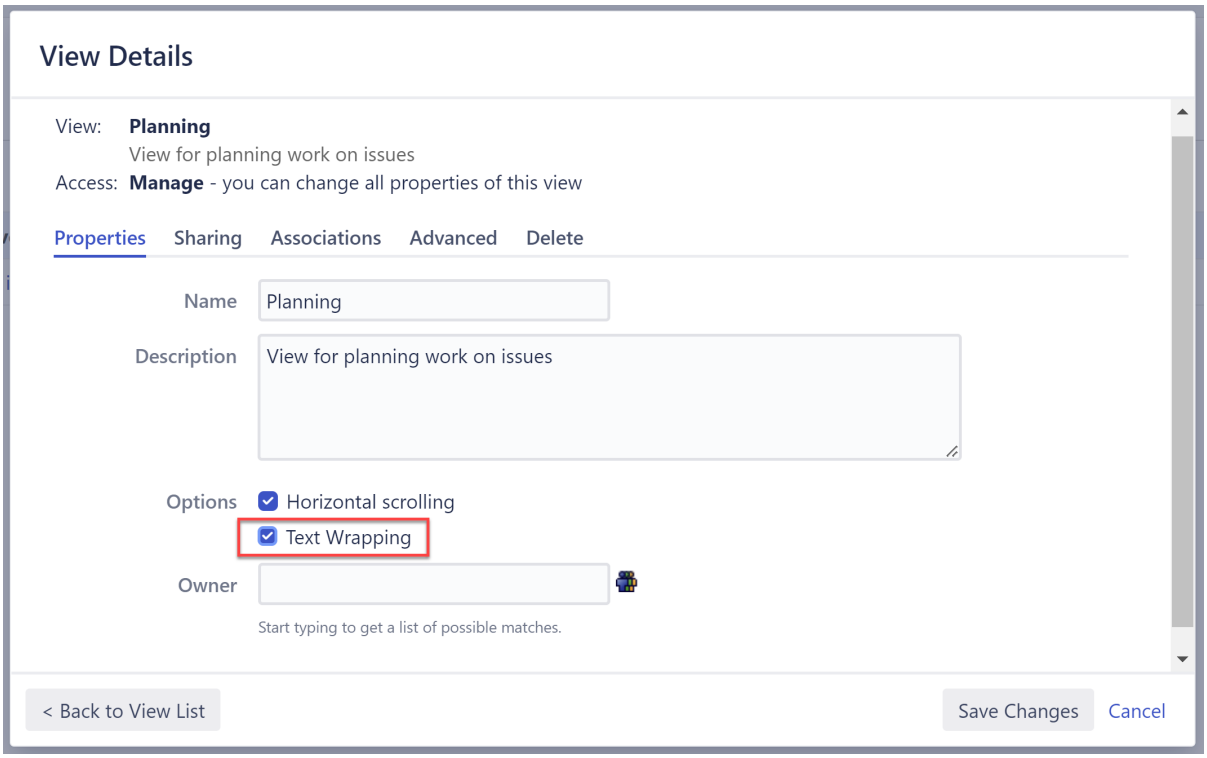


3.10.9.2 Saving Text Wrapping Settings

Structure allows you to save your text wrapping settings for a particular view.

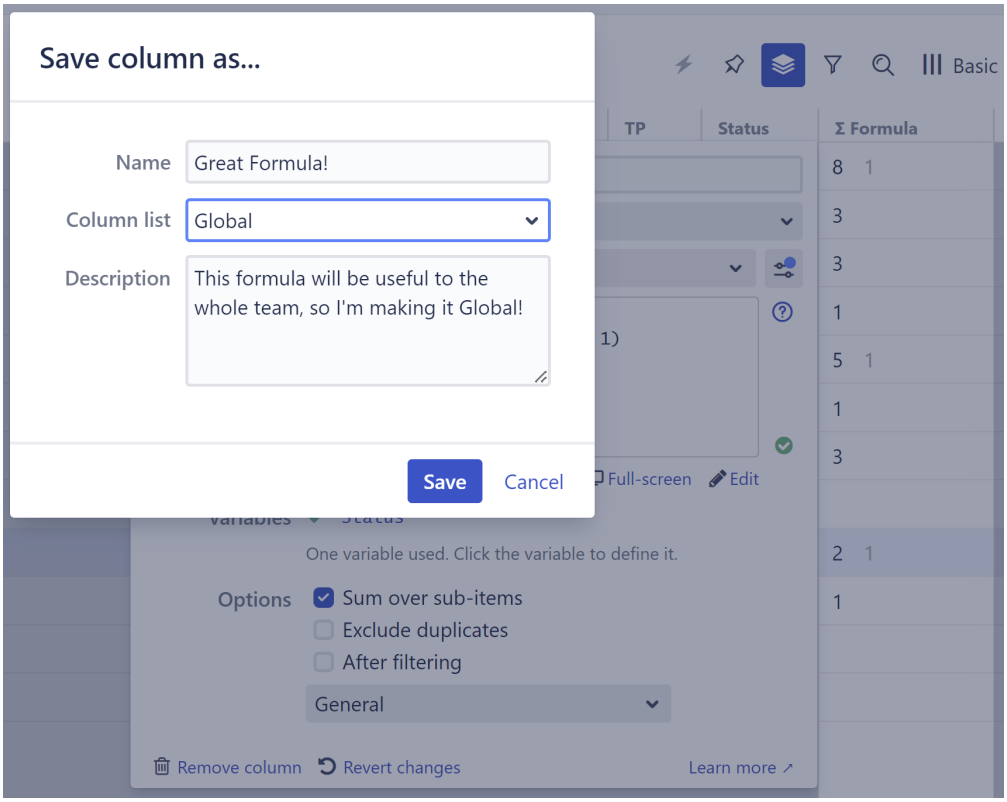
To set your text wrapping preference for a saved view:

- In the Views Menu, select **Manage Views**
- Locate the view you want to set and click **Details**
- Under the Properties tab, check or uncheck **Text Wrapping**



3.10.10 Saved Columns

Saved Columns allow you to save, reuse, and share custom formula columns - and even use them in generators.



Saved Columns can be used within the same structure, in new structures, and even by other members of your team (if they're saved as Global saved columns).

3.10.10.1 Learn More

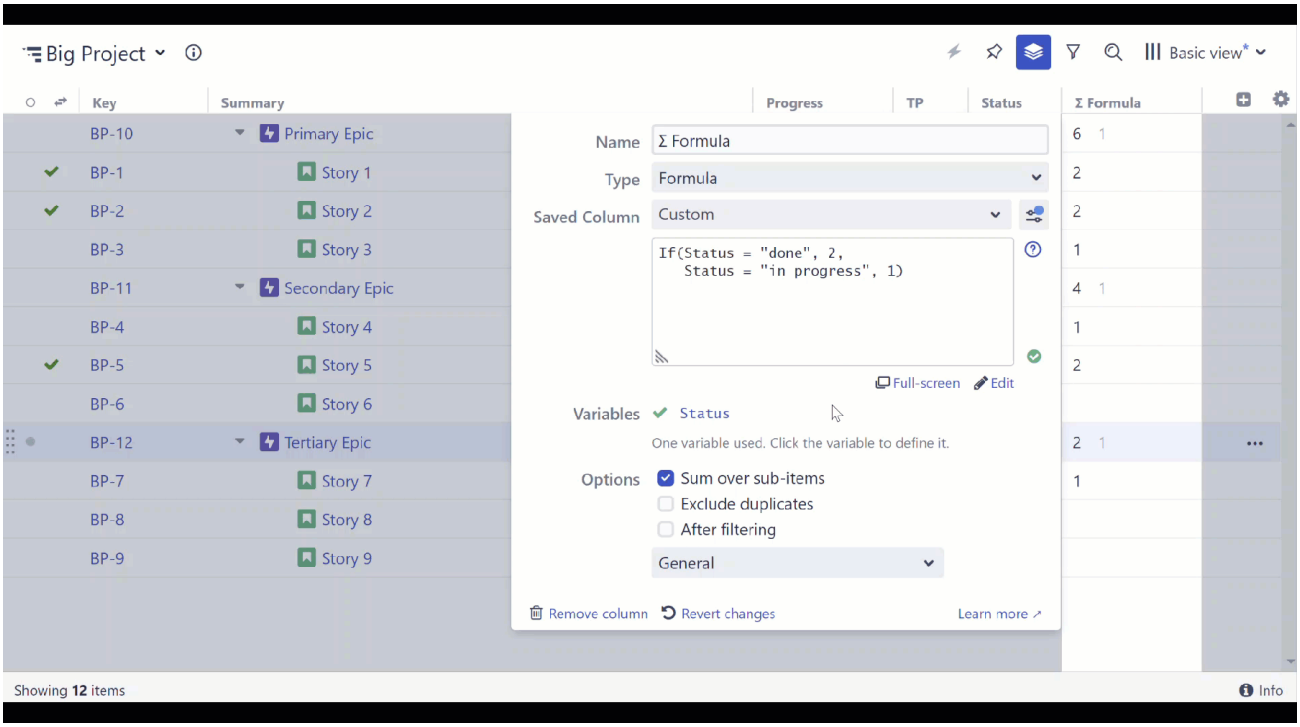
- Saving a Formula as a Saved Column(see page 525)
- Sharing a Saved Column(see page 526)
- Opening a Saved Column(see page 527)
- Modifying a Saved Column(see page 528)
- Using a Shared Column in a Generator(see page 529)

3.10.10.2 Saving a Formula as a Saved Column

Once you've created a formula you want to save and/or share, you can turn it into a Saved Column, which can be opened and modified in any structure.

To save a formula:

1. Complete your formula and make sure it works as expected.
2. Click the settings button beside the Saved Column selector.
3. Select **Save as...**
4. Enter a name and description for your formula column.
5. Choose whether the column should be private or global (available to anyone with Structure access).
6. Click **Save**.

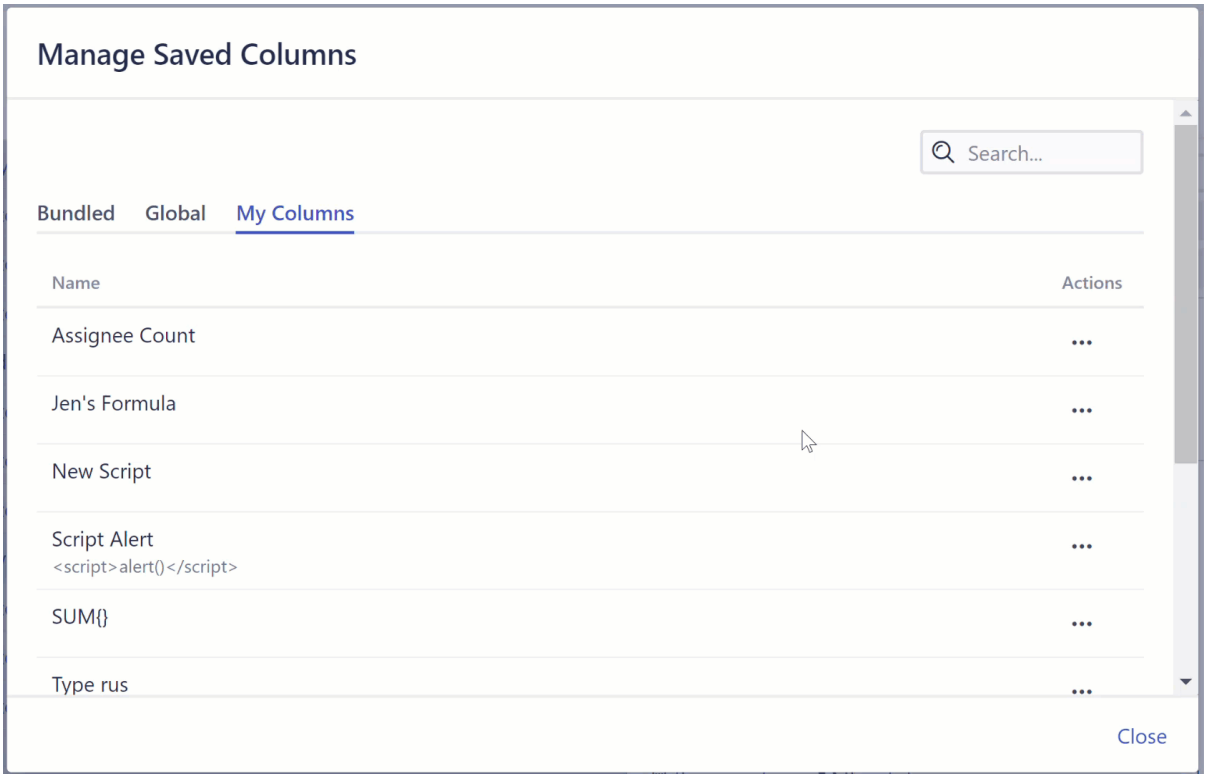


3.10.10.3 Sharing a Saved Column

To share a Saved Column, select the **Global** column list when saving the column.

If you've already saved the column:

1. Open Manage Saved Columns(see page 530).
2. Locate the column you want to share.
3. Click the three dots in the Actions column.
4. Select **Move to Global**.
5. Click **Proceed** to verify the move.

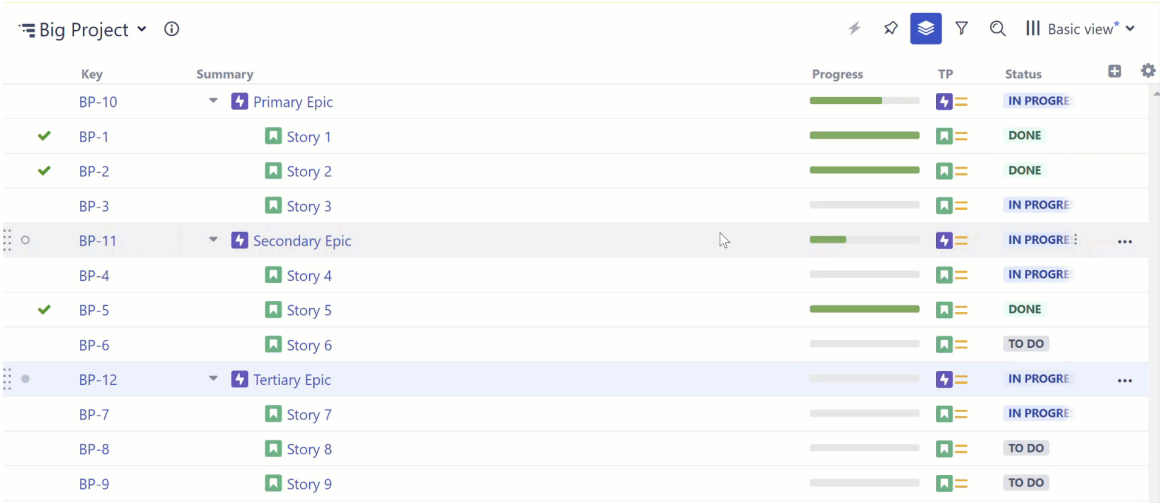


Manage Global Saved Columns permission(see page 1028) is required to save a column to the Global list. You can If you don't have this permission, speak with your Structure administrator.

3.10.10.4 Opening a Saved Column

To open an existing Saved Column:

1. Add a new Formula column using the Add Column menu
2. Open the Saved Column dropdown
3. Scroll through the list of saved columns, or use the search bar. *Note: if you don't find it, check the other lists (Bundled, Global, My Columns).*
4. Select your saved formula column.

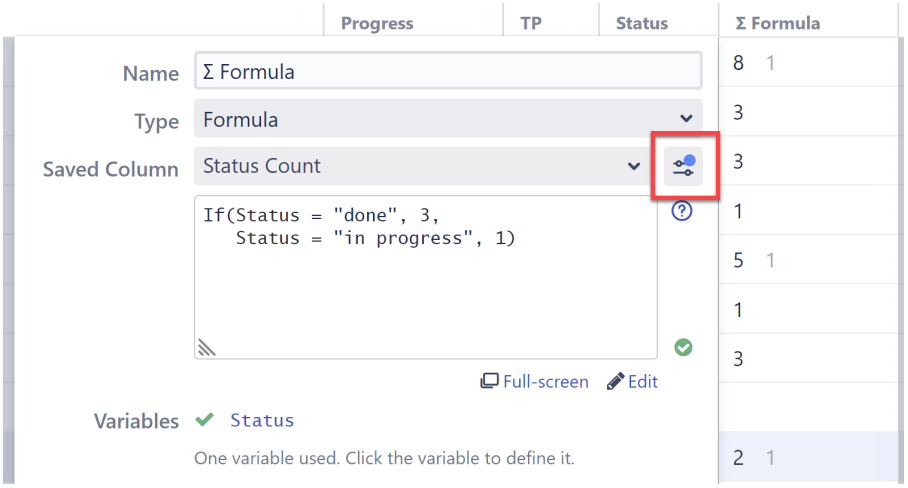


You can now use and even customize the formula. If you make changes to the formula, those won't be seen by anyone else using the formula unless you save them.

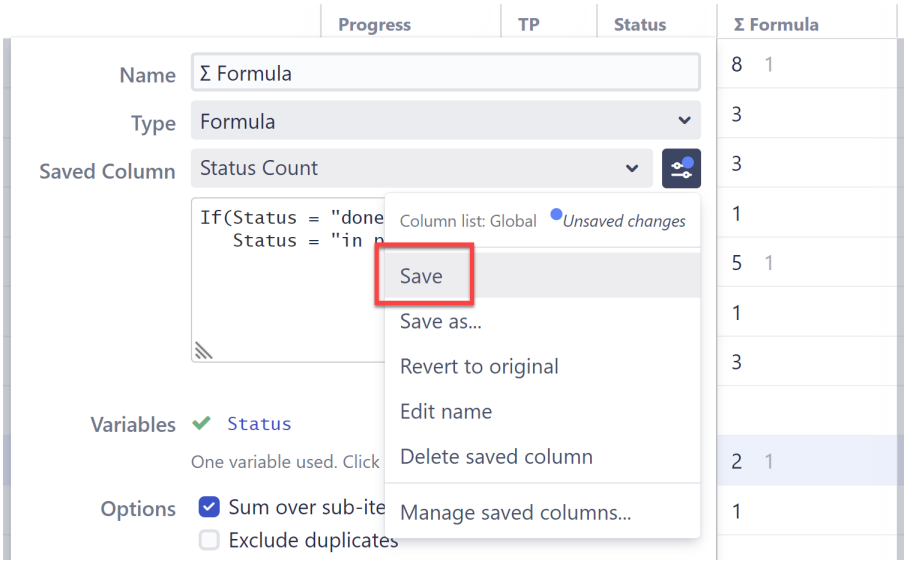
3.10.10.5 Modifying a Saved Column

Once you open a Saved Column, you can make changes to it just as you would any other formula column. Those changes are local (not visible to anyone else) and do not affect the original Saved Column - until you save them.

Once you make changes to a Saved Column, you'll see a blue indicator on the Saved Column settings button.

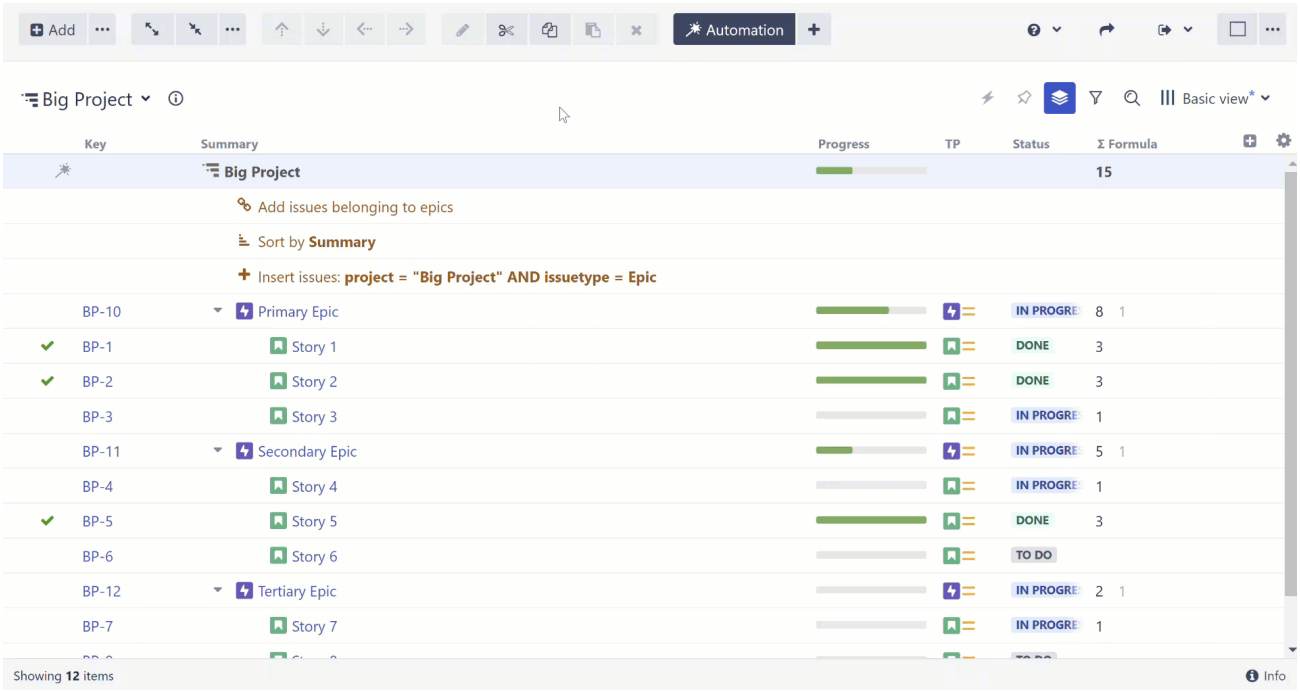


To update the saved file with those changes, click the settings button and select **Save**.



3.10.10.6 Using a Shared Column in a Generator

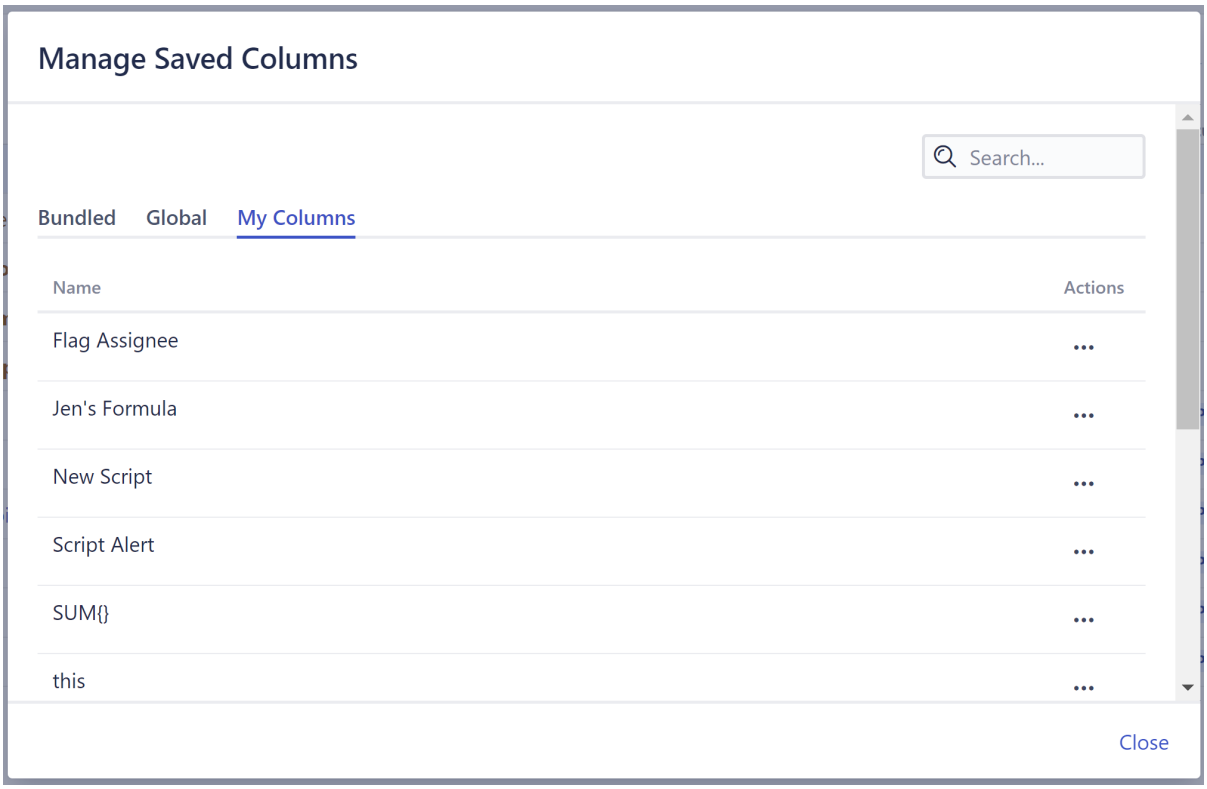
Saved columns can be used in any generator that allows you to use formulas. For example, the Filter by Attribute generator:



The generator creates a copy of the saved column's formula, so any changes made to the saved column after the generator is created will not affect the generator. To apply those changes, you will need to create a new generator.

3.10.10.7 Managing Saved Columns

To edit, copy, share, or delete a Saved Column, open the Manage Saved Columns screen.

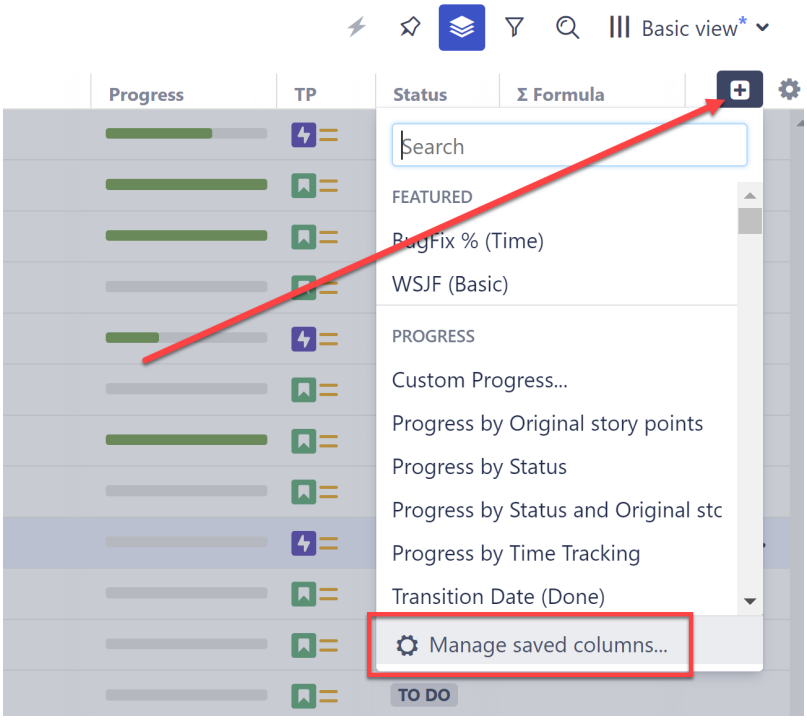


There are two ways to open the Manage Saved Columns screen:

- From the Add Column menu(see page 530)
- From a Formula column(see page 0)

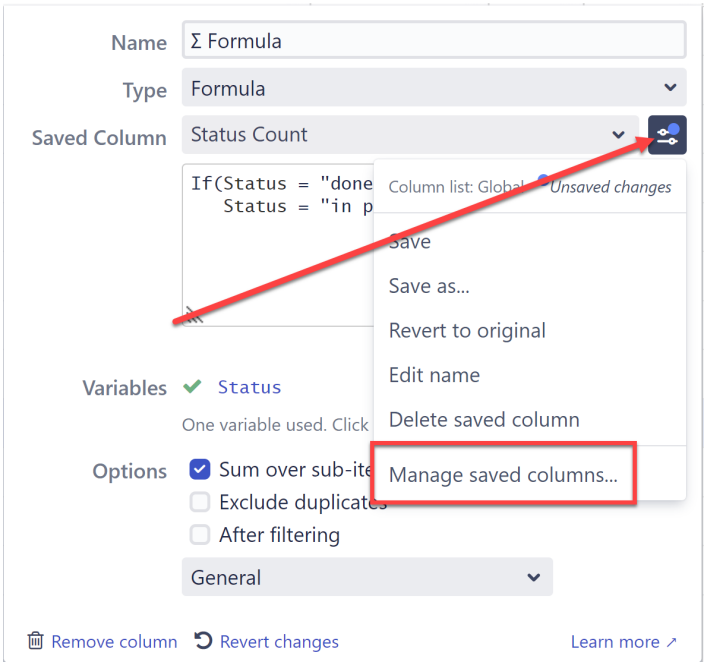
From the Add Column Menu

From the main structure window, click the + icon to open the Add Column menu. At the very bottom, click **Manage saved columns...**



From a Formula Column

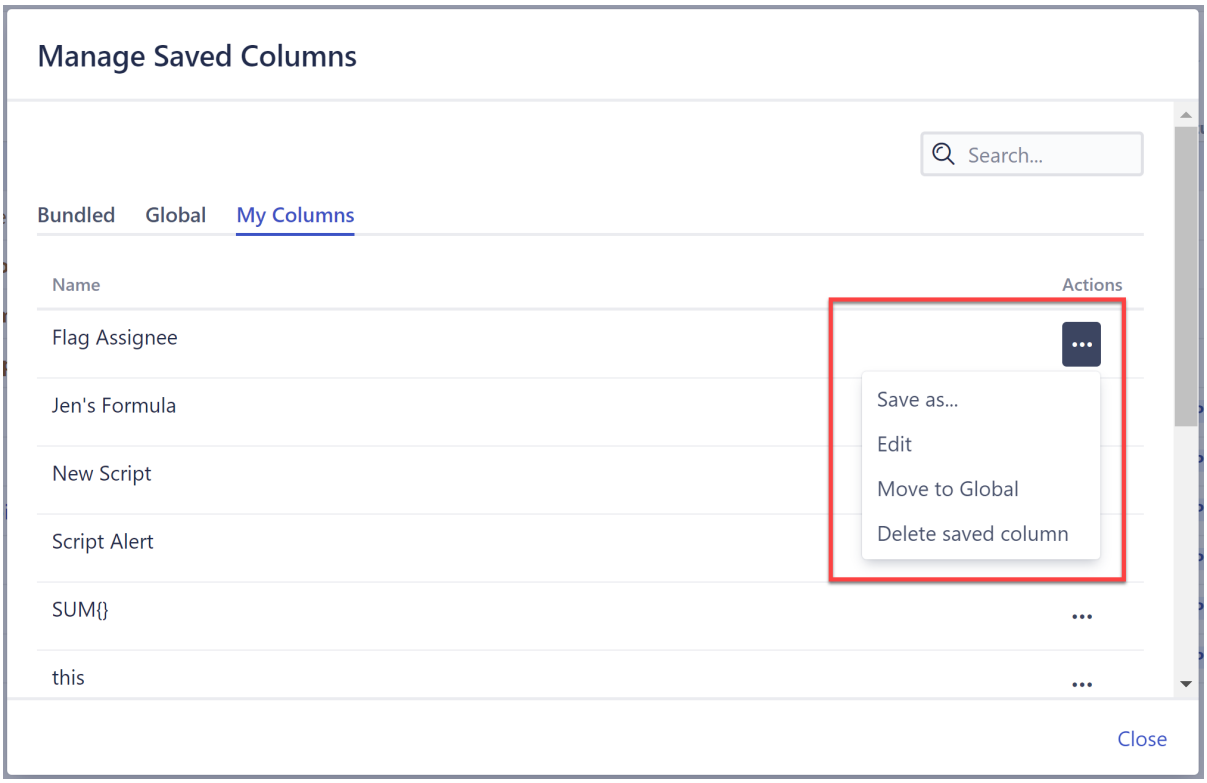
From any Formula column, click the Saved Column settings button and select **Manage saved columns...**



Managing a Saved Column

From the Manage Saved Columns screen, locate the saved column you want to manage, click the three dots in the Actions column, and select one of the following actions:

- Save as... - make a copy of the saved column
- Edit - edit the column's name or description
- Move to Global - make the column available to everyone on the Jira instance
- Delete saved column - permanently delete the column

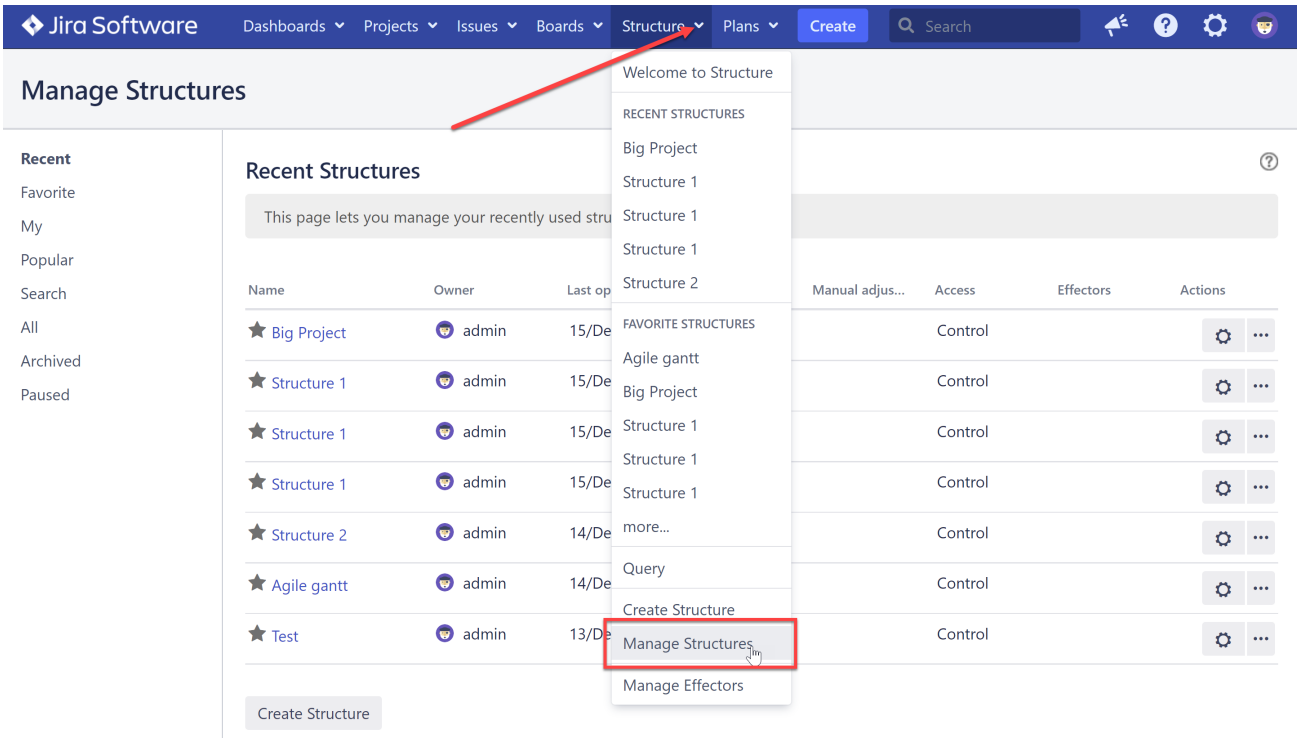


⚠ Manage Global Saved Columns permission(see page 1028) is required to move a saved column to the Global list. If you don't have this permission, speak with your Structure administrator.

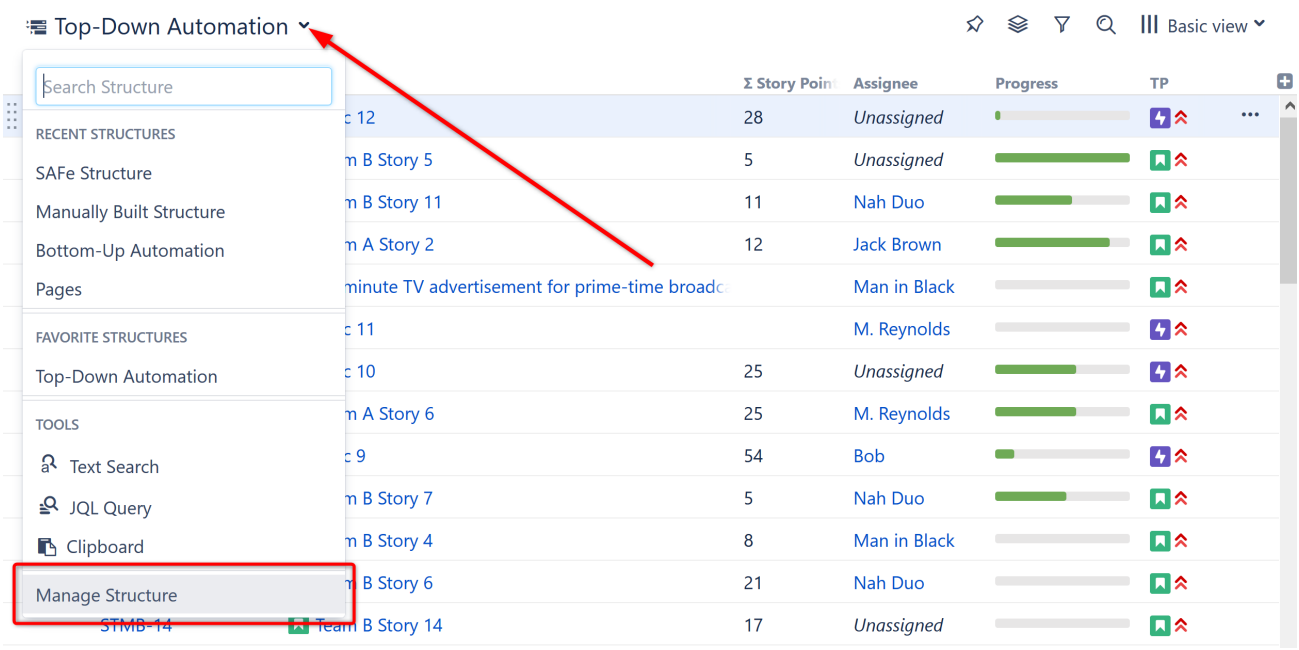
3.11 Managing Structures

The Manage Structures page lets you view, search for, create, and delete structures, as well as change their settings.

To open the Manage Structures page, go to the **Structure** menu in the top navigation bar and select **Manage Structures**.



You can also reach the Manage Structures page from the structure selector menu:



The **Manage Structures** page contains the following tabs:

- **Current** – shows the structure you are currently working with
- **Recent** – lists recently viewed structures, starting with the most recent/current structure
- **Favorite** – lists structures that you have marked as your favorite(see page 537)
- **My** – lists structures created by you

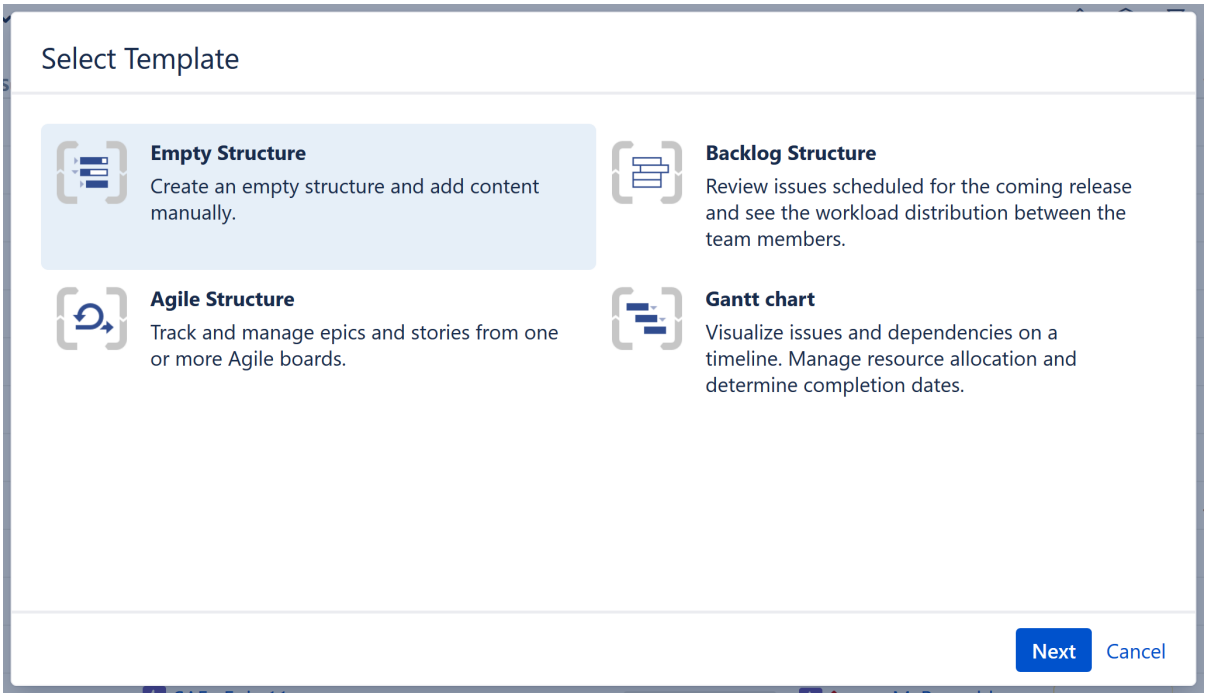
- **Popular** – lists structures that are marked as favorite by at least 2 users, ordered by their popularity(see page 537)
- **Search** – allows you to find a structure by name, owner or ID(see page 536)
- **All** – lists all structures visible to you
- **Paused** – only displayed when there are structures with Paused Automation(see page 532)

Since anonymous users cannot create structures or mark structures as their favorites, **Favorite** and **My** tabs are not shown when you are not logged in.

Learn more about managing structures:

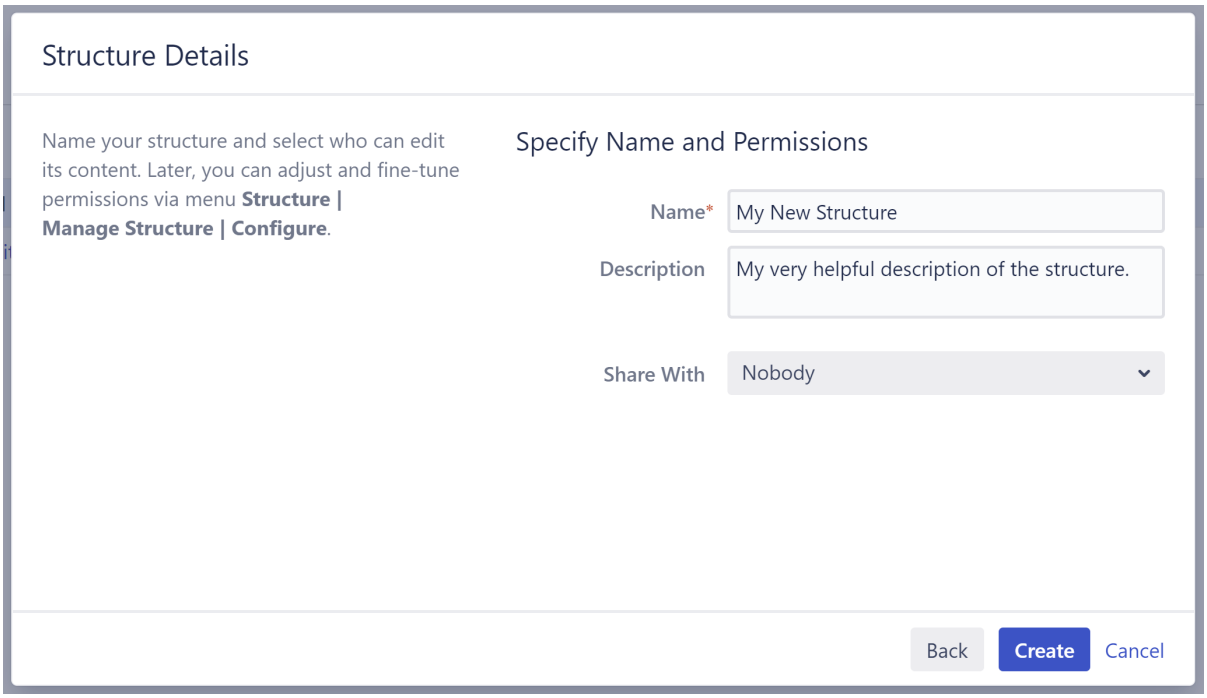
3.11.1 Creating New Structures

To create a new structure, select **Structure | Create Structure** in the top menu or click the **Create Structure** button on the Manage Structures(see page 532) page. You have the option of using one of our template wizards to streamline the creation of your new structure, or starting with an empty structure.



When creating a new structure, you must specify at least the structure's name. You can optionally add:

- Description - Enter a clear description of the structure. This is displayed the first time a user opens the structure, so anyone you share it with can easily understand its purpose.
- Share with - Select users to share the structure with. These users will have Edit permissions, but you can change this setting or add addition permissions later using the Structure Details(see page 539) page.



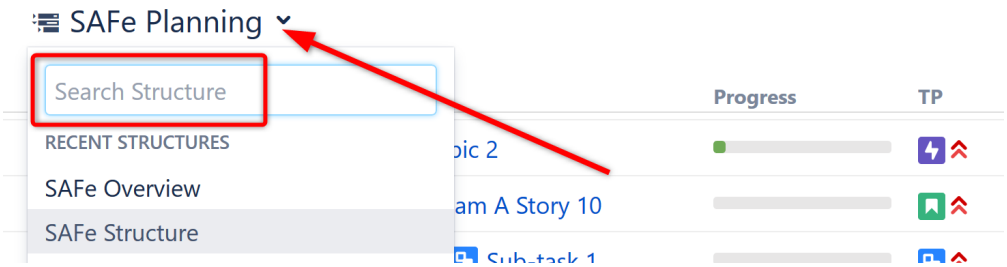
> When you create a new structure, you become the owner of the structure. Structure owner always has full access to the structure - see Structure Permissions(see page 543).

If you choose one of our templates, the template wizard will ask you a series of questions to help you add and organize issues to your specific needs. If you choose an empty structure, you can add (see page 102)and organize(see page 108) issues or apply automations(see page 140) once the structure is created.

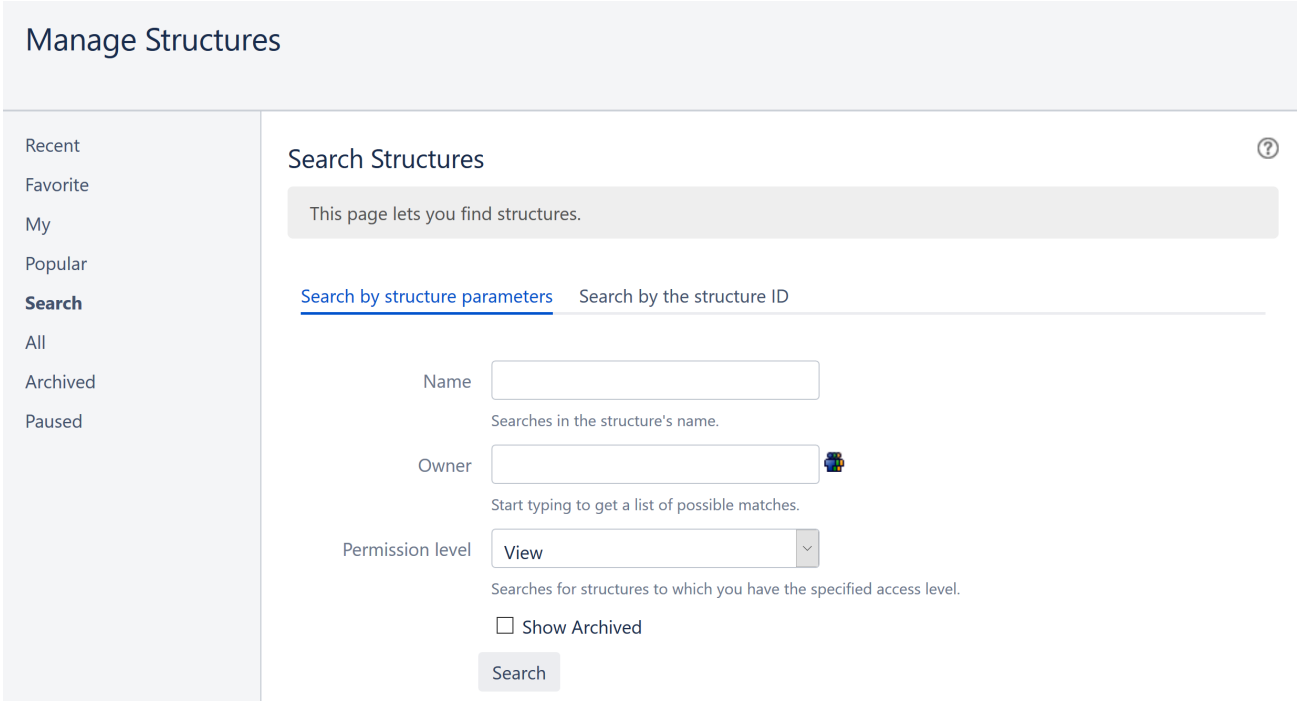
> Only logged-in users who have access to Structure are allowed to create new structures. See Who Has Access to the Structure(see page 955) for more details.

3.11.2 Locating a Structure

The easiest way to locate a structure is to click the name of the current structure and search for the new structure in the drop-down search box.



If you don't know the exact name of the structure you're looking for, or you need to do more than simply open the new structure, you can use Manage Structure(see page 532). From the Jira menu, go to **Structure | Manage Structures** and select the **Search** tab.



3.11.2.1 Finding Structures by Name, Access Level or Owner

To search for structures by their properties:

1. Enter any of the search parameters. Parameters are:

Name	Only structures that contain the specified text in their name will be shown. If you don't know the exact name, you can use a part of a word that you know should be in the structure's name.
Owner	Only structures that are owned by the specified user will be shown. *Note: This only works if you have permission to browse users.*
Permission Level	Lets you select the structures that you can Edit or Control, according to the selected permission level. (For example, if you select **Edit** permission level, you will see all structures that you can edit and control, but you will not see structures that you can only view.)

2. Click **Search** without entering any parameters. All structures visible to you will be shown.

3.11.2.2 Finding a Structure by Its ID

To perform a search by a structure's numeric ID:

- Click the **Search by the structure ID** tab.
- Enter the structure ID. (It must be a number.)
- Click **Search**. If there's a structure with the specified ID and you have the permission to view it, it will be shown.

3.11.3 Default Structure

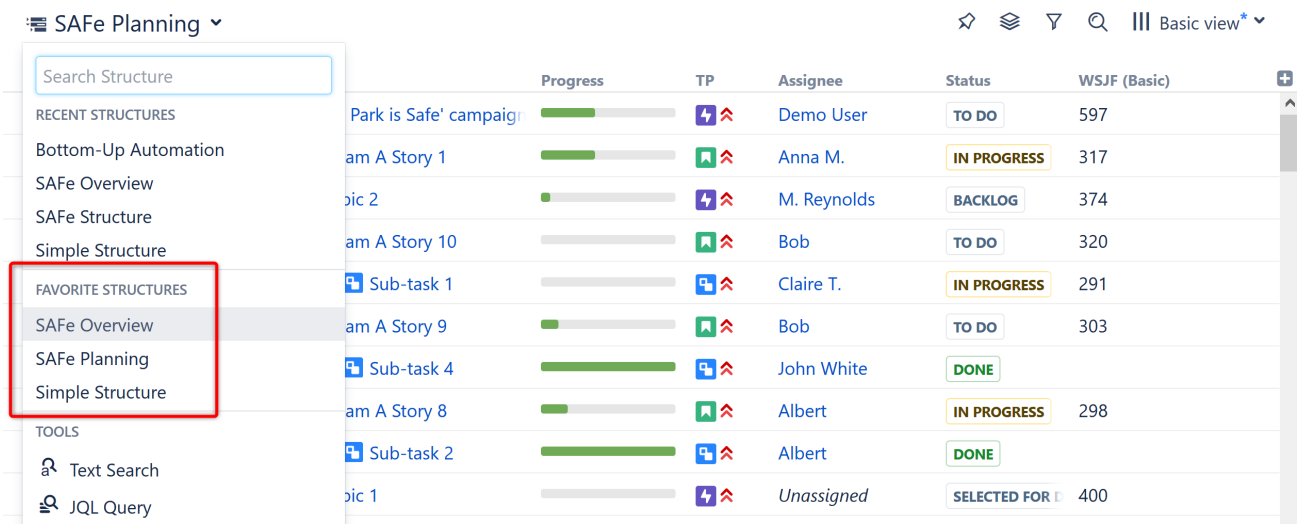
A default structure is displayed when no specific structure has been selected, typically the first time a user opens Structure. The default structure may also be displayed on Issue Pages(see page 93) and Project Pages(see page 95) when the **Auto-switch to default structure** option is selected.

Default structures can be set by:

- Jira administrators can change the system default structure(see page 958)
- Project managers can select a project-level default structure(see page 0) for any project that is enabled for Structure

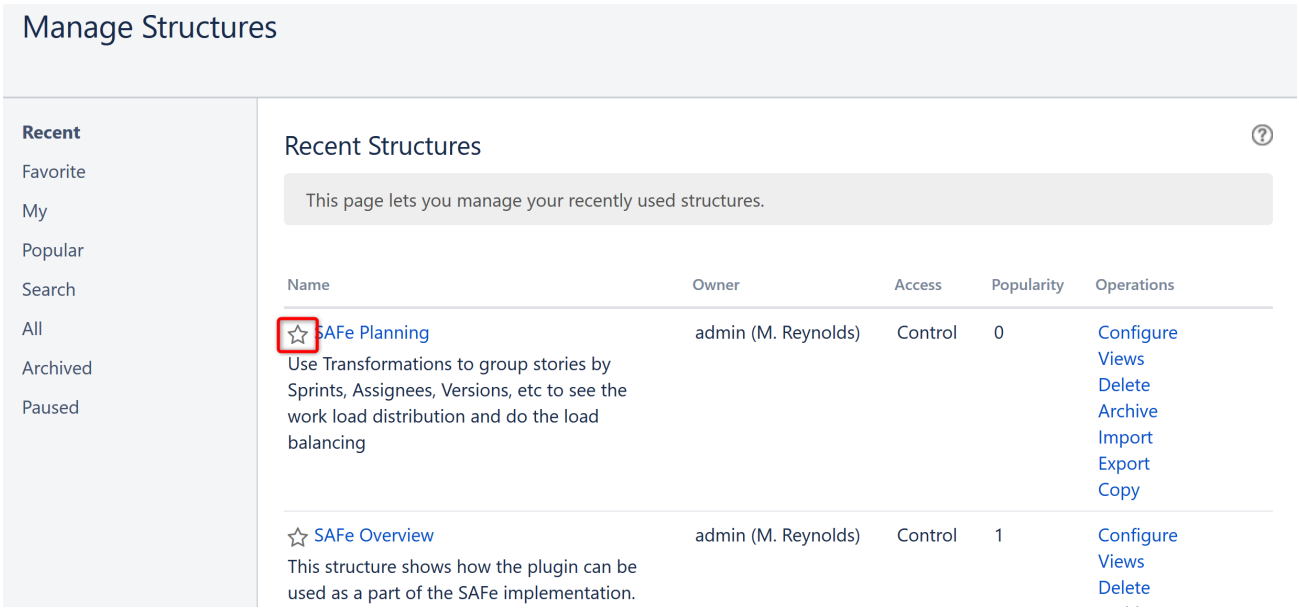
3.11.4 Favorite Structures

You can favorite structures for quick access. When you click the current structure's name, the Favorite Structures section of the drop-down list displays the top 5 favorite structures ordered alphabetically by name.



3.11.4.1 Add/Remove Favorite Structures

To favorite a structure, open Manage Structures(see page 532) and locate the structure(see page 536) you want to favorite. Click the white star (☆) near the name of that structure. The star will then be shaded (★) to indicate that the structure was added to your favorite structures list.



To remove a structure from your favorites list, simply click the star again. To view all favorited structures, open Manage Structures(see page 532) and click the Favorite tab.

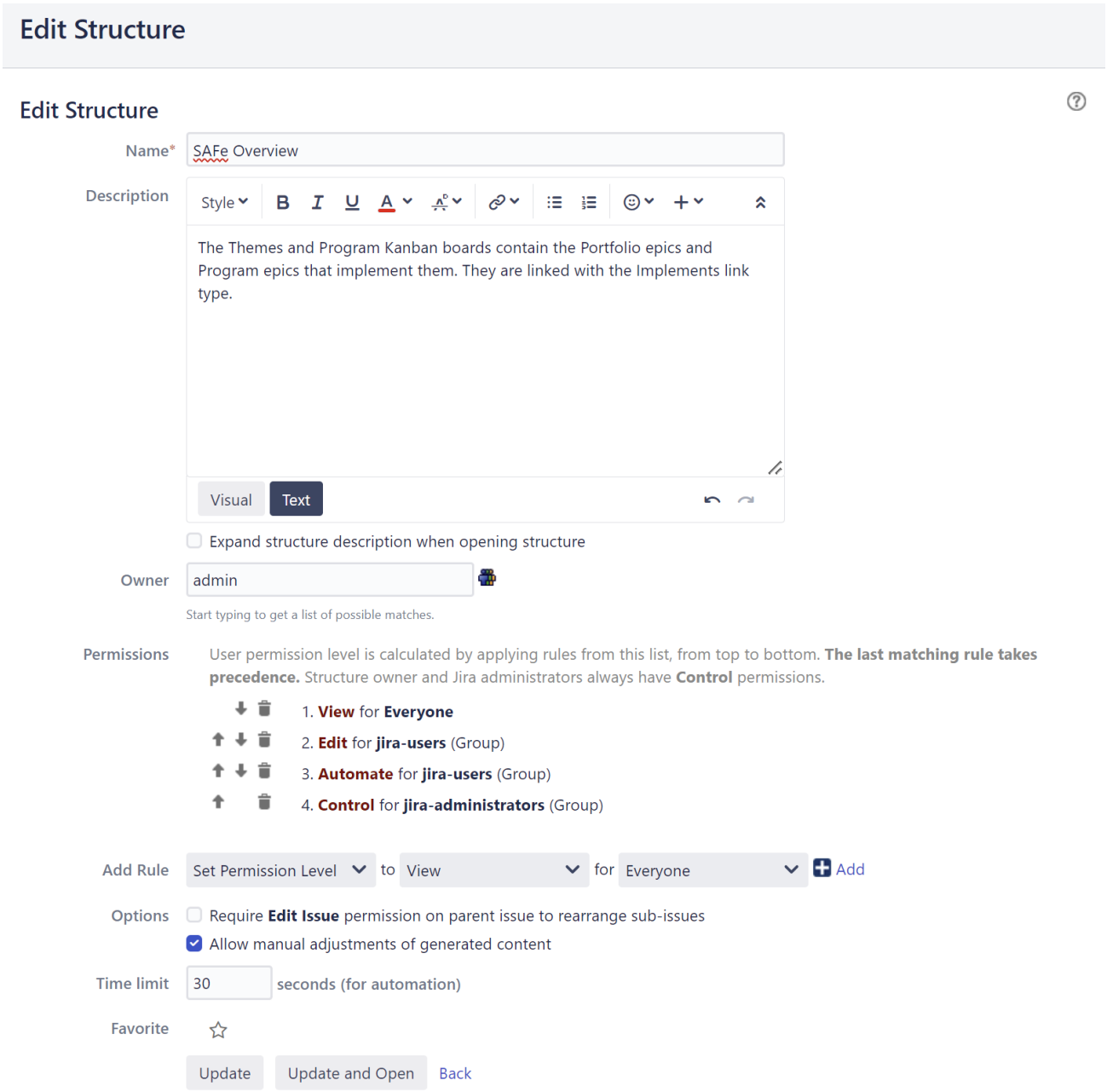
3.11.4.2 Structure Popularity

Structure **popularity** is the number of users who have marked this structure as a favorite. To view the most popular structures, open Manage Structures(see page 532) and click the **Popular** tab.

3.11.5 Structure Details

Every structure has the following parameters:

- **Name** (*required*) - Name is used to identify the structure in the drop-down menus, including the *Structure* menu in the top navigation bar.
- **Description** - Used to describe the purpose of the structure to users.
    - *Expand structure description when opening structure* - When set, the description will be displayed the first time a user opens the structure.
- **Owner** - The owner of the structure. *To change the owner, you must have Control permission for the structure.*
- **Permissions** - Defines who can view, edit or configure the structure. See Structure Permissions(see page 543) for details.
- **Optional Settings:**
    - *Require Edit Issue permission on parent issue to rearrange sub-issues* - When set, users cannot move sub-issues unless they have permission to edit the parent issue as well. See Structure Permissions(see page 543) for details.
    - *Allow manual adjustments of generated content* - Enables or disables manual adjustments.(see page 194)
- **Time Limit** - Determines the maximum amount of time Automations(see page 140) can run before being paused(see page 201).
- **Favorite** - When selected (the star is filled in), the structure will appear in your Favorite Structures(see page 537) list.

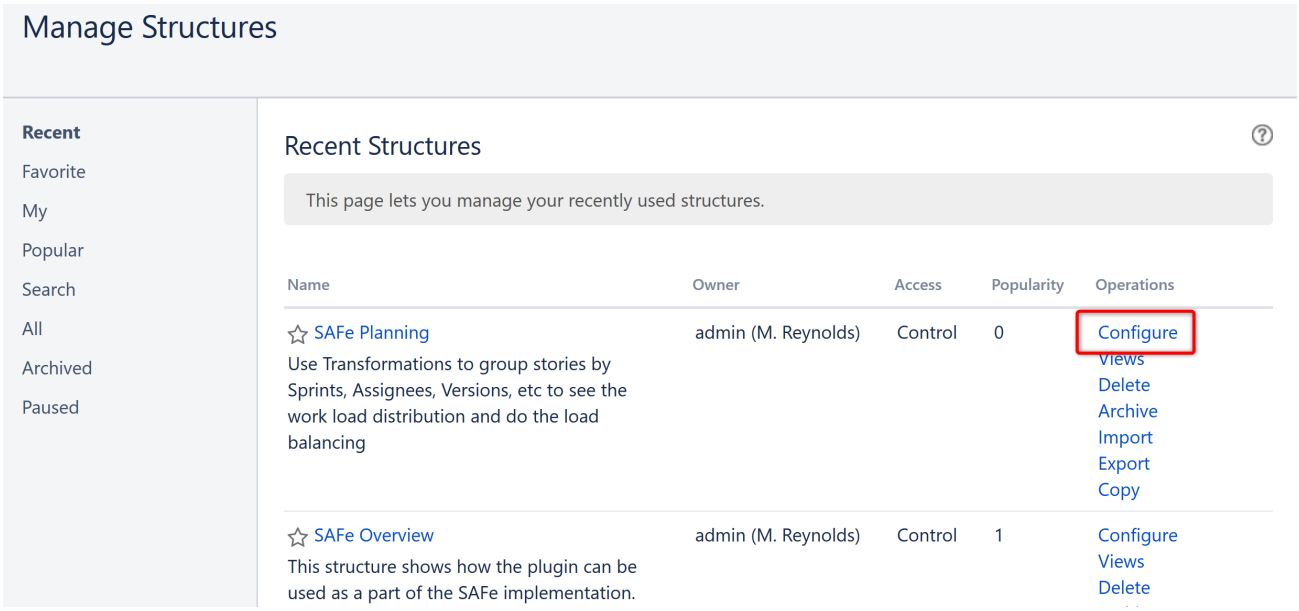


You can specify structure details when Creating New Structures(see page 534) and when Editing Structure Details(see page 540).

3.11.5.1 Editing Structure Details

The Edit Structure screen allows you edit details(see page 539) of a structure. To access the Edit Structure screen:

1. Open the **Structure** menu and select **Manage Structures**.
2. Locate the structure you need to change and click the **Configure** link in the **Operations** column.

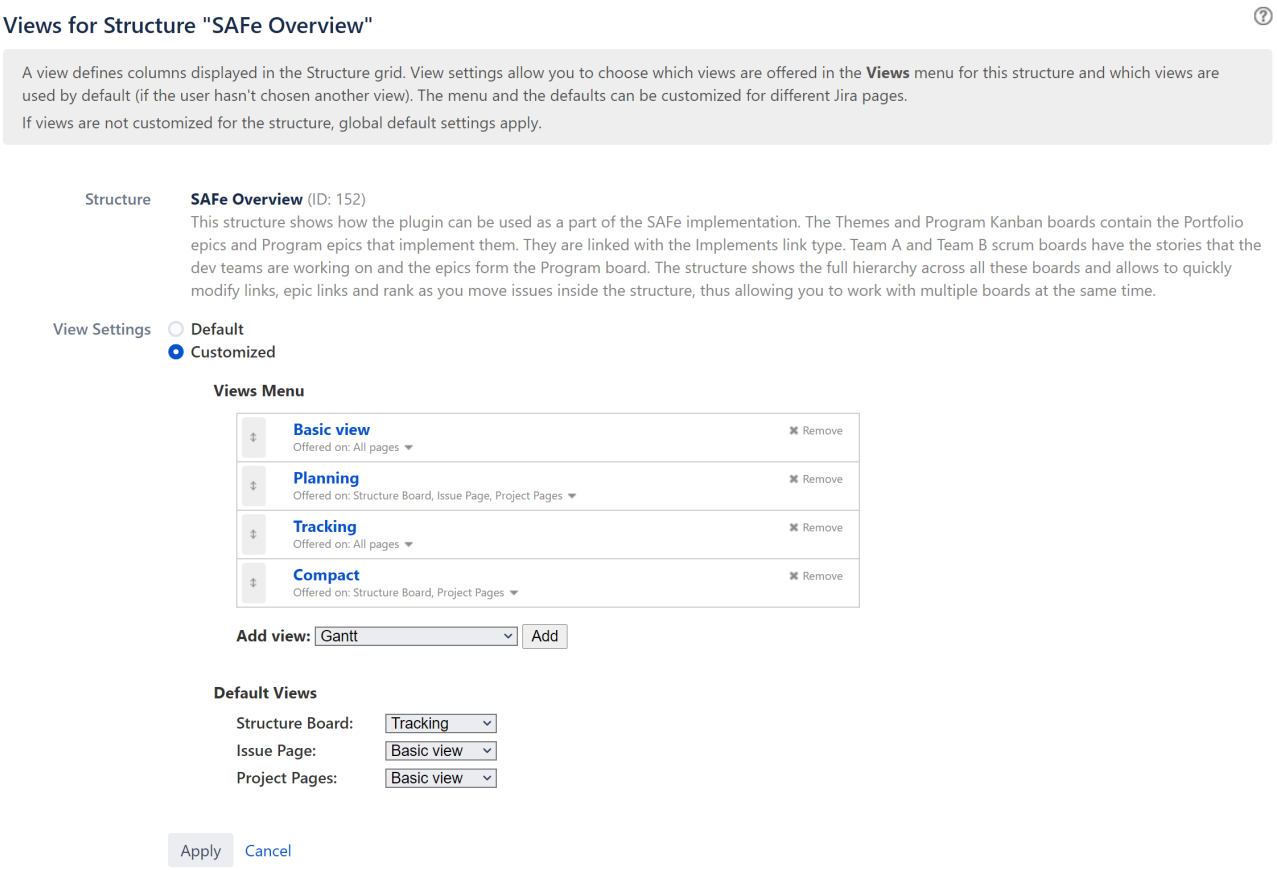


If you do not see a **Configure** link, you probably do not have Control permission for that structure.

See Structure Details(see page 539) for more information.

3.11.6 Customizing View Settings

A structure's view settings determine which views are offered to the users in the Views Menu(see page 507) and which view is used by default. To customize view settings for a structure, open the Manage Structures (see page 532)page, locate the structure and click the **Views** link.



You must have **Control** permissions for the structure to adjust its view settings.

3.11.6.1 Switching Between Default and Customized View Settings

Initially, each structure uses the default view settings, defined globally for all structures. To customize view settings for the structure, select the **Customized** radio button. The default settings are copied and you can adjust them to your needs.

To revert to the default view settings, select the **Default** radio button.

You can change the default global view settings if you are a Jira administrator – open the **Administration | Structure | Defaults** tab and click **Change** in the Default View Settings section.

3.11.6.2 Configuring Views Menu

The **Views Menu** section lets you select which views appear under the Associated Views section of the Views Menu(see page 507) for each type of Jira page(see page 88).

- To add a view to the menu, select the view in the **Add view** drop-down and click **Add**
- To remove a view from the menu, click the **Remove** button
- To change a view's position in the menu, drag the view by the drag handle at the left of the view bar

- To restrict a view's appearance in the menu to some specific pages, click the **Offered on:** line and select the pages where you'd like this view to be used

A view is **associated** with a structure if it is part of the Views Menu, as defined by the structure's view settings.

3.11.6.3 Configuring Default View

In the **Default Views** section, you can select which of the views included in the Views Menu section is the default view for a given Jira page (Structure Board, Structure Board with Issue Details, Issue Page and Project Page). Pick one view from those offered in the drop-down for each page.

If the **Views Menu** does not include any views for a specific page (for example, no views for Issue Page), you won't be able to configure the default view for that page.

Changes take effect when you press **Apply**.

3.11.7 Structure Permissions

Every structure has a list of permission rules, which define who is allowed to see, edit or configure the structure.

3.11.7.1 Access Levels

Each user has one of the following access levels to a structure:

None	The user does not see the structure at all and does not know that it exists.
View	The user can view the structure but cannot make changes.
Edit	The user can view the structure and can rearrange, add and remove issues from the structure. The user cannot, however, create or modify generators[83].
Automate	The user has full edit access to the structure, including modifying generators[84] and effectors(see page 179).
Control	The user can view, edit and configure the structure - including changing structure permission rules.

83 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4
84 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4

3.11.7.2 Default Access

By default, all users have **None** access level.

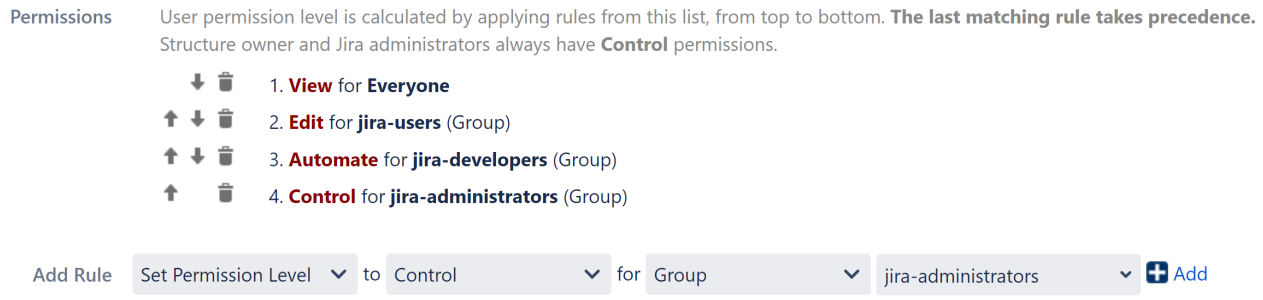
The structure's owner and Jira administrators always have **Control** access level.

Therefore, if you create a new structure and do not specify any permission rules, it will be a private structure that only you and Jira administrators will be able to see and modify.

3.11.7.3 Permission Rules

Users who have **Control** permission for a structure can define permission rules by Editing Structure Details(see page 540).

The Permission Rules list is an ordered list that's used to calculate the access level for a given user. Each rule assigns an **access level** to a specific **condition** (category of users).



The conditions are applied from top to bottom, and the **last matching rule has precedence** - so if a user fits multiple conditions, their access level will match the lowest-listed matching condition. For this reason, it is advisable to order permissions from least access (None) to most access (Control), as we've done above.

Setting Permissions

To set permissions, in the **Add Rule** section, select an access level and choose one of the following conditions for that access level:

Everyone	Matches any user, including anonymous (not logged in). This condition can be used to set a default permission for everyone.
Group(G)	Matches users that belong to the group G.
Project Role(R,P)	Matches users that have role R in project P.
User	Adds the permission for a specific Jira user.

By default, you will only be able to edit permssions for groups you belong to. To set permissions for other groups, speak with your Jira admin.

Admins can update this behavior by changing the `structure.permissions.allowAllUserGroups` property in the Structure Dark Features and Fine Tuning Interface(see page 974).

To copy the permissions from another structure, click the **Set Permission Level** box and choose **Apply Permissions From**. This will let you apply the same permissions rules from any structure for which you have Control access level.

Permission Examples

The following are examples of how your permissions list might look:

- Everyone can view the structure, Jira administrators can edit, only the owner and admins can control:

1. **View** for **Everyone**
2. **Edit** for **jira-administrators** (Group)

- Any logged-in user can edit the structure, except for the users from the structure-noaccess group, who can't even view the structure. Project administrators are allowed to control the structure:

1. **Edit** for **jira-users** (Group)
2. **None** for **structure-noaccess** (Group)
3. **Control** for **Administrators** of **SAFe Program** (Project Role)

- Incorrect configuration - in this example, everyone is given View access level:

1. **Control** for **jira-developers** (Group)
2. **Edit** for **jira-users** (Group)
3. **View** for **Everyone**

  Although the configuration looks fine at first glance, remember that **the last matching rule has precedence**. So even if a user is part of the jira-developers or jira-users group, their access level will be set to View by the last rule.

Require Edit Issue permission on parent issue

When this option is selected on the Structure Details(see page 539) page, the user must have Edit Issue permission for a parent issue in order to adjust its sub-issues. In other words, direct sub-issues (or children issues) are treated as if they are part of the parent issue; therefore, adding sub-issues, removing sub-issues and rearranging sub-issues is actually changing the parent issue - for which the Edit Issue permission is required.

The user must also have **Edit** access level to the structure to be able to make changes at all.

Note the following:

- Top-level issues do not have parent issues, and therefore are not affected by this flag: the user can add/ rearrange issues at the top level of the structure if they have Edit access level.

- The Edit Issue permission applies only to the direct parent issue. If issue A has sub-issue B, and B has sub-issue C, then to be able to move or remove C from the structure, the user needs Edit Issue permission on B - not on A.

Structure maintains a cache of users permissions with regards to each structure. In most cases, the cache is recalculated automatically, but in some cases Structure may miss a change in a user's groups or roles. This could mean that the changed permissions for the user do not take effect until several minutes later (but only with regards to Structure Permissions(see page 543)).

A user can force the cache to be recalculated by doing a **hard refresh** from the browser. Typically, it's done by holding **Ctrl** or **Shift** or both and clicking the **Refresh** button.

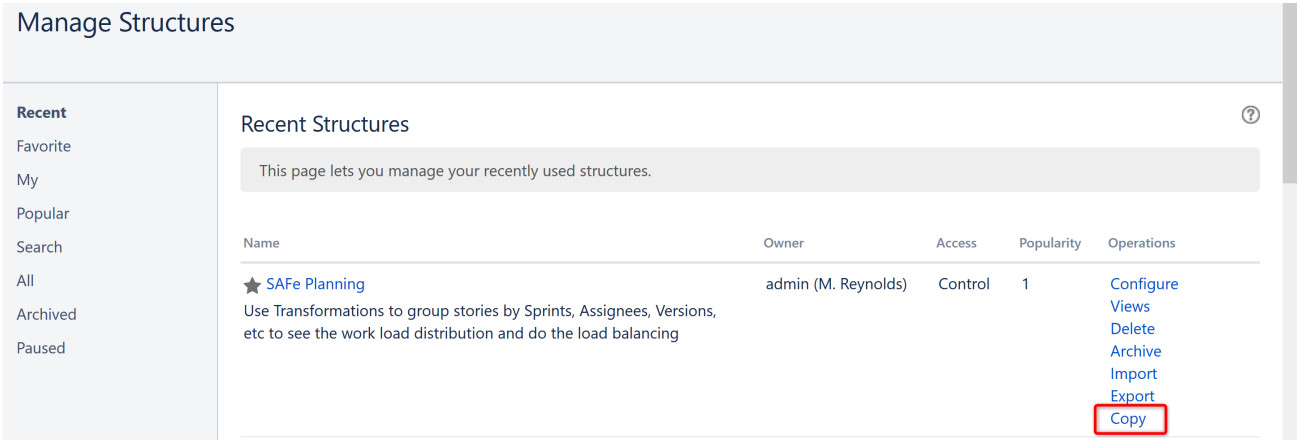
3.11.8 Copying a Structure

With the **Copy** action, you can create a full copy of a structure, and, optionally, clone every issue in the structure.

If you need to copy only a part of a structure, create a new empty structure and use Issue Clipboard(see page 111) to copy a part of the structure.

3.11.8.1 Create a Copy

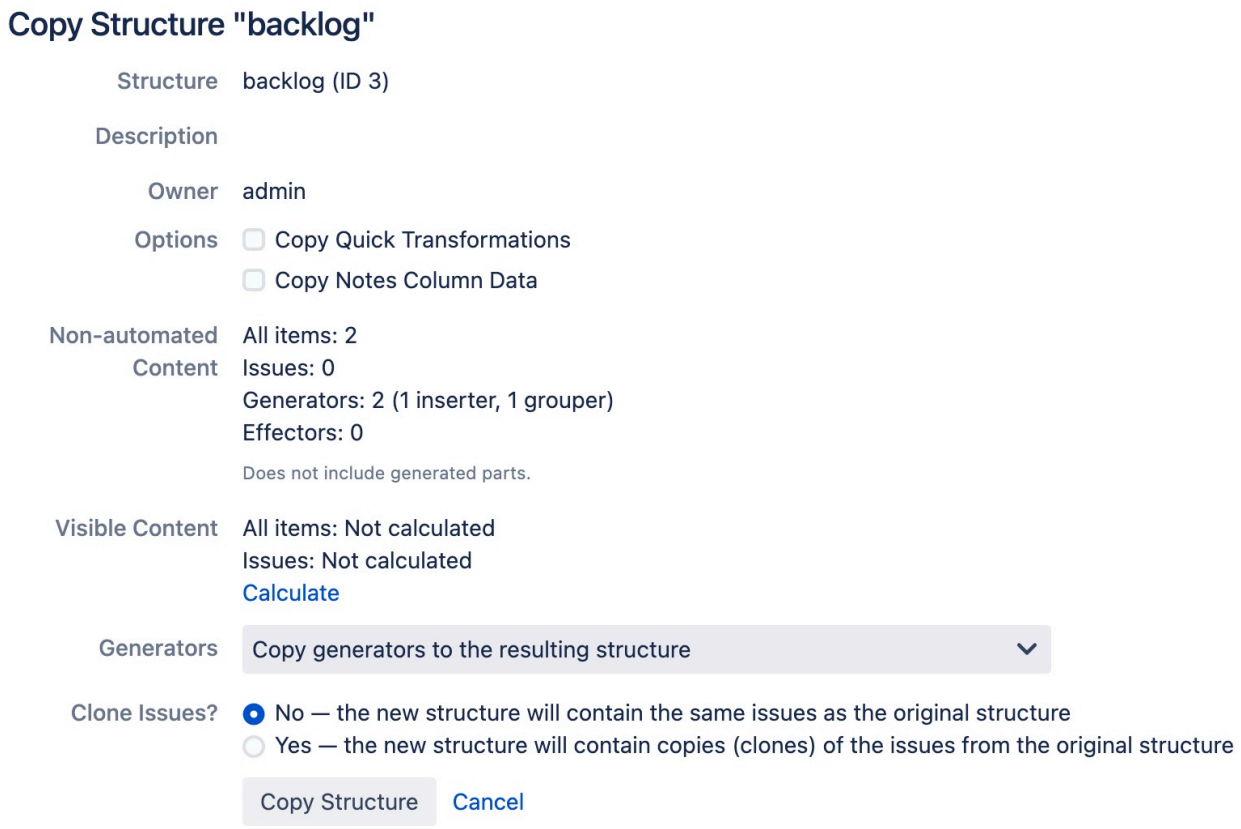
To create a copy of a structure, open the Manage Structures(see page 532) page using the top navigation **Structure** menu. Find the structure you'd like to copy and click the **Copy** link in the Operations column.



If you don't see **Copy** in the Operations column, you probably do not have permissions (see page 955)to create new structures.

The **Copy Structure** page will show you the information about the structure, including its size and the number of issues, generators[85], and synchronizers(see page 983) it contains. If the structure contains automation(see page 140), you can click the **Calculate** link in the **Visible Content** section to execute the generators and see the generated content statistics.

85 https://wiki.almworks.com/display/structure/.Types+of+Generators+v7.4



Select the following options for copying the structure:

1. **Choose whether or not to include Quick Transformation and/or Notes column data.**
2. **Choose how to handle generators and generated content**, if the structure has any:
   - Copy generators to the resulting structure – the new structure will contain copies of the generators from the original structure, which will generate the same content. This is the default.
   - Replace all generators with the generated content – the new structure will contain only the generated content (issues, folders, etc.) and not the generators themselves (so the content will not automatically update with Jira changes).
   - Do not copy generators – the new structure will contain only the non-automated content from the original structure. The generators and generated parts will not be copied.
3. **Choose if you'd like to copy** synchronizers(see page 983), if the structure has any. If you don't see a **Copy Synchronizers** option, either the structure does not have any synchronizers or you do not have permission(see page 991) to create synchronizers. See Copying Synchronizers(see page 1007) for more details.
4. **Choose if you'd like to clone issues** (see Copy vs. Clone(see page 548)). If yes, enter the additional parameters for the cloning process(see page 548).

Once you've made you selections, press **Copy Structure** or **Start Cloning**.

3.11.8.2 The New Structure

The new structure is created with the following properties:

- Structure name is automatically set to "Copy of <*old structure name*> (<*date of copy*>)".
- Structure description is copied.
- View settings are copied.

- You become the owner of the copied structure.
- If you have **Control** access level to the original structure, permission rules are copied. Otherwise, permission rules for the new structure are empty (it is a private structure). To share the new structure, add permission rules(see page 543).

You can immediately edit the new structure's properties on the screen with the copy result.

3.11.8.3 Copy vs. Clone

When copying a structure, you can choose to have the new structure use the same issues as the original structure or clone the original issues (create copies). The following chart explains the differences:

	Copy Structure	**Copy Structure & Clone Issues**
Selected answer for **Clone Issues?**	**No**	**Yes**
New structure created?	Yes	Yes
New structure contains:	same Jira issues as the original structure	clones (copies) of the issues from original structure
Quick?	Yes	No, a background process is launched to do issue cloning
Permissions required:	**View** access to the original structure **Create Structure** permission	**View** access to the original structure **Create Structure** permission **Bulk Change** global Jira permission A number of project-level permissions

For details about configuring and running cloning, see Copying Structure and Cloning Issues(see page 548).

3.11.8.4 Copying Structure and Cloning Issues

When copying a structure(see page 546), you can turn on the **Clone Issues** parameter and have Structure create a copy (clone) of every issue in the original structure.

How Issue Cloning Works

Each issue in the original structure is cloned by creating a new issue with the same:

- summary
- description
- field values, including custom fields

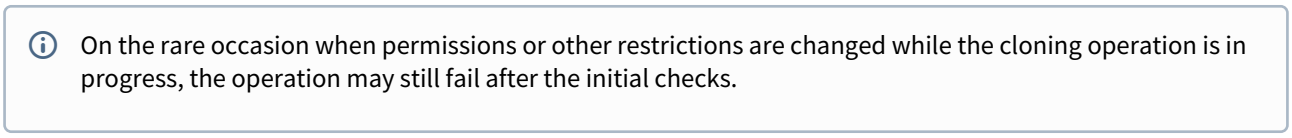
With the following exceptions:

- The **Status** field is not copied. The cloned issues are always created in the initial status, according to each issue's project and workflow scheme.
- If a field is not present on the Create Issue screen, its value is not copied. The cloned issues will have the default value for that field instead.
- Archived versions are removed from **Affects Versions**, **Fix Versions**, and custom fields that have versions as values.

Additionally, when cloning issues to a different project:

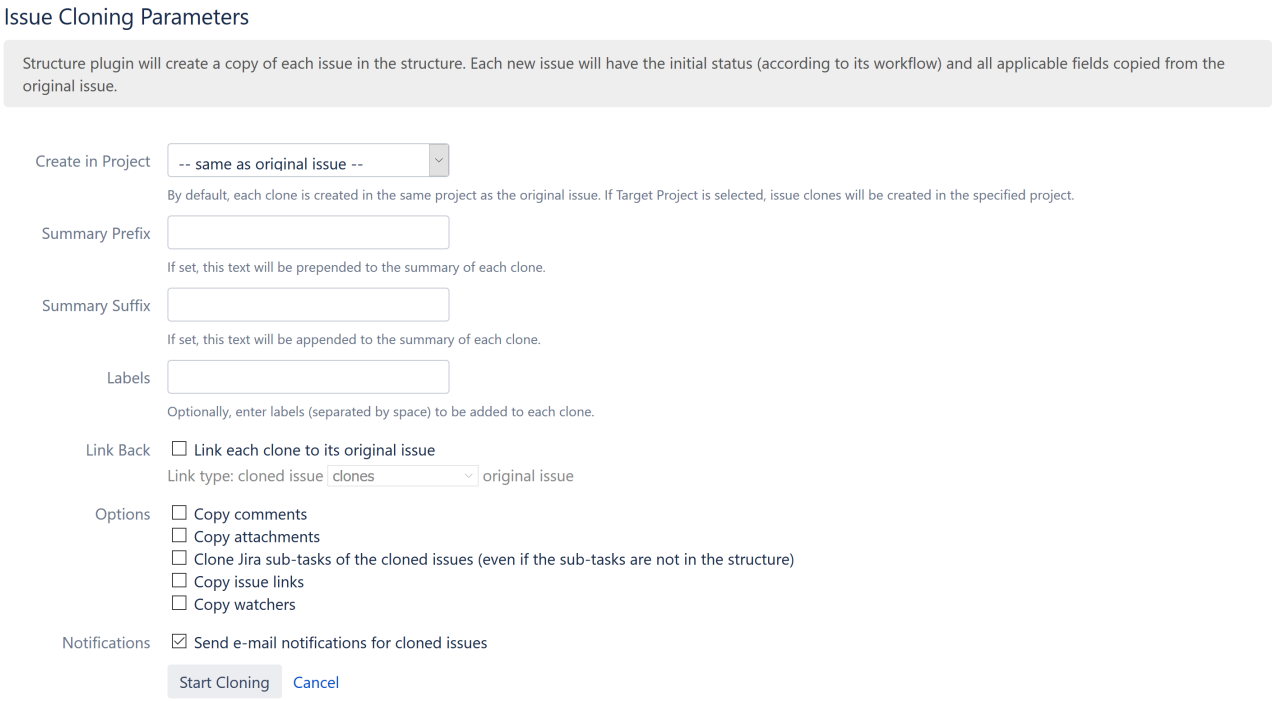
- If custom fields for the original issues are not available in the new project, the values of those custom fields are not copied.
- If field values of the original issues are not available in the new project, those values are removed. For example, this may happen to the **Components** field, or to fields that take versions as values.
- In some cases, cloning issues to a different project may not be possible. For example, when a certain field is required in the target project, but absent (or not required) in the source project. If this is the case, you will need to either change the target project restrictions or make sure every issue in the copied structure satisfies them.

Structure does its best to verify that it can clone each issue in the original structure **before** it begins the actual cloning. If Structure detects a potential data loss (for example, because one of the custom fields is absent in the target project), it warns you and lets you decide whether you want to continue. If no issues can be cloned (for example, if you do not have **Create Issues** permission for the new project), the operation stops before creating any clones.



Cloning Parameters

When Clone Issues is selected, you can specify a number of additional parameters.



Create in Project	Lets you specify a project for the new issues, different from the project the issues currently belong to. If not specified, every new issue is created in the same project as the original issue.
Summary Prefix **Summary Suffix**	Lets you modify the summary of the clones. If the resulting summary gets longer than the Jira limit (255 characters), it will be truncated.
Labels	Lets you add space-delimited labels to the cloned issues. (Already existing labels are preserved.)
Link Back	If specified, every new issue will be linked with its original issue.
Copy comments	If selected, all comments are also copied. If not selected, new issues will have no comments.
Copy attachments	If selected, attachments are copied (the actual files are copied on the Jira server).

Clone Jira sub-tasks of the cloned issues	If selected, when issues have sub-tasks in Jira which are not also in the structure, those sub-tasks will be cloned as well. For example, issue A-1 has sub-task A-2 in Jira. A-1 is in the structure being copied, but A-2 is not. With this option selected, A-2 will also be cloned. Otherwise, A-1 will be cloned without the sub-task.
Copy issue links	If selected, all issue links and remote issue links will be copied. If a link exists between two issues, and both are cloned, then the new link will be created between the clones. ⚠ If you use the **Link Back** option, then the links of the type selected for linking back to original issues will **not be copied**. Scrum board Epic-Story relationships are also copied when you select this option. The rule is the same as for issue links: • If you clone an epic together with its stories, the cloned stories will be added to the cloned epic. • If you clone the stories alone, the clones will be added to the original epic.
Copy watchers	If selected, the users watching an original issue will be added to the watcher list of the clone.
Notifications	If selected, an email may be issued for every created issue, depending on the Jira notification scheme for each issue's project.

Required Permissions

The following permissions are required to copy a structure with issue cloning:

- To be able to clone structure issues, you need **Bulk Change** global permission.
- Because the result of cloning is a new structure, you also need to be allowed to create new structures. (Configured by the Jira administrator - see Administrator's Guide(see page 955).)
- You need to have **Create Issue** permission in the projects where the clones are created. If you specify the **Create in Project** option, the issues will be created only in the specified project. Otherwise, clones are created in the same projects as their respective original issues.
- Users in the **Assignee** field of the original issues will have to have **Assignable User** permission in the target project – otherwise, cloned issues cannot be assigned to that user and will be assigned by default.
- If you don't have **Modify Reporter** permission, you won't be able to set the value of the **Reporter** field in the cloned issues. Instead of the original reporter, you will be the reporter of the issue clones.
- You need to have **Add Comments** permission to copy comments, **Link Issues** permission to copy issue links or use **Link Back**, **Create Attachments** permission to copy attachments, **Manage Watchers** permission to copy watcher lists, and **Edit Issue** permission to copy Epic-Story relationships.

Executing Bulk Cloning

When you press the **Start Cloning** button, a background process starts on the Jira server, which performs the following:

1. Copies the original structure's hierarchy and stores it in memory
2. Checks all necessary permissions required for cloning
3. Clones all issues
4. Creates a new structure and fills it with the cloned issues

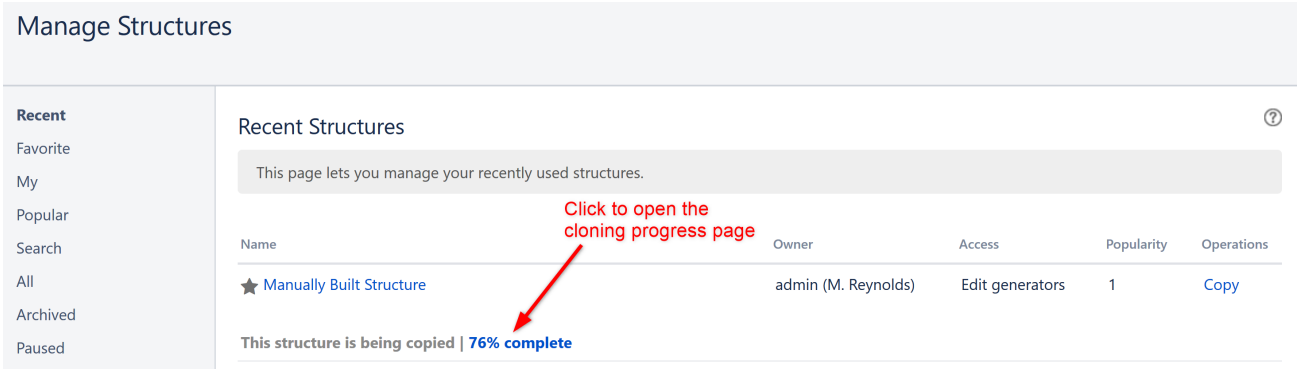
At step 2, the cloning process might discover some problems. If critical problems are discovered, an error message is shown and the process is aborted. If non-critical problems are discovered, warnings are shown and user input is required. The warnings may suggest that cloning may continue, but the resulting issues might not be exact copies. After your confirmation, the process continues.

As cloning proceeds, a progress bar is shown on the screen. When cloning is done, the Edit Structure page is opened for the resulting structure, so you can make any necessary adjustments to its name and permissions.

Cloning issues is potentially a long operation. Cloning a structure with tens of thousands of issues may take an hour or more. Cloning smaller structures should take considerably less time.

Checking Cloning Progress

When cloning has started, you can navigate away from the cloning progress page. To see the progress and get back to the progress screen, open the Manage Structures(see page 532) page and locate the original structure. It should show that the structure is being copied.



When cloning is completed, or if there are warnings or questions from the cloning process, the link will read "Waiting for input". Click the link to open the cloning progress page.

Cancelling Cloning

You can cancel the cloning process from the cloning progress page by pressing the **Cancel** link.

Issues that have already been created by the cloning process will be assembled into a special structure named "*[Cancelled Cloning Result]*". You can use Bulk Change(see page 138) to quickly delete the unwanted issues.

Cloning Queue

Cloning issues can place considerable load on a Jira server. To avoid overloading the server with cloning jobs, there is a limit to the number of cloning processes that can happen simultaneously. If this limit is exceeded, your cloning process will initially be in "waiting" state, pending for other cloning processes to finish.

3.11.9 Archiving a Structure

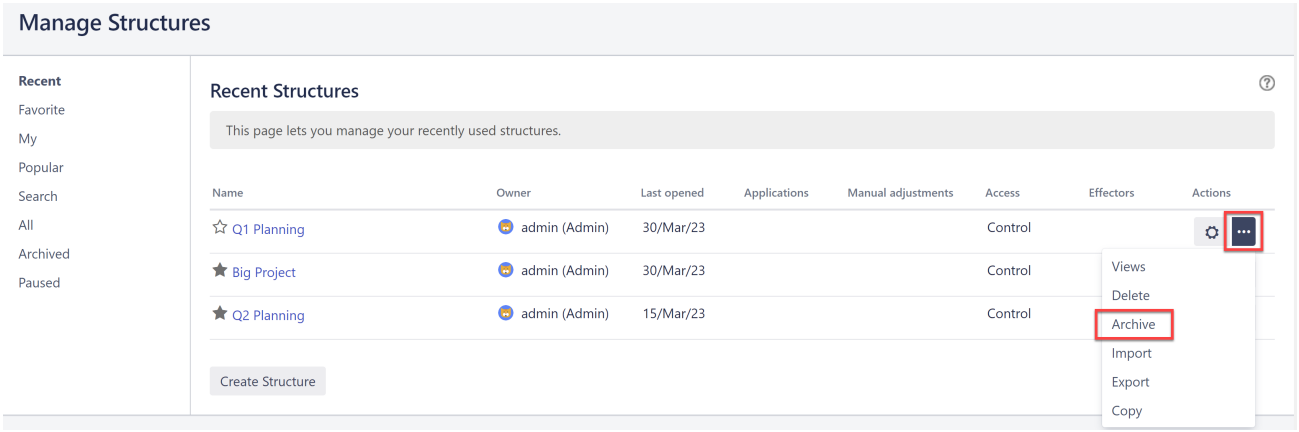
When you **Archive** a structure it becomes read-only and is hidden from search results and menus. The issues within the structure are not affected in any way. They remain in Jira and can still be part of another structure.

> *Read-only* means that users cannot add, remove or move items (issues, folders, memos, generators) in the archived structure.

3.11.9.1 Archive a Structure

To archive a structure:

1. Open the Manage Structures(see page 532) page using the top navigation **Structure** menu.
2. Find the structure you'd like to archive, open the Actions menu (...), and select **Archive**.
3. Review the structure you are about to archive and confirm the operation. You can **Unarchive** the structure in the future.



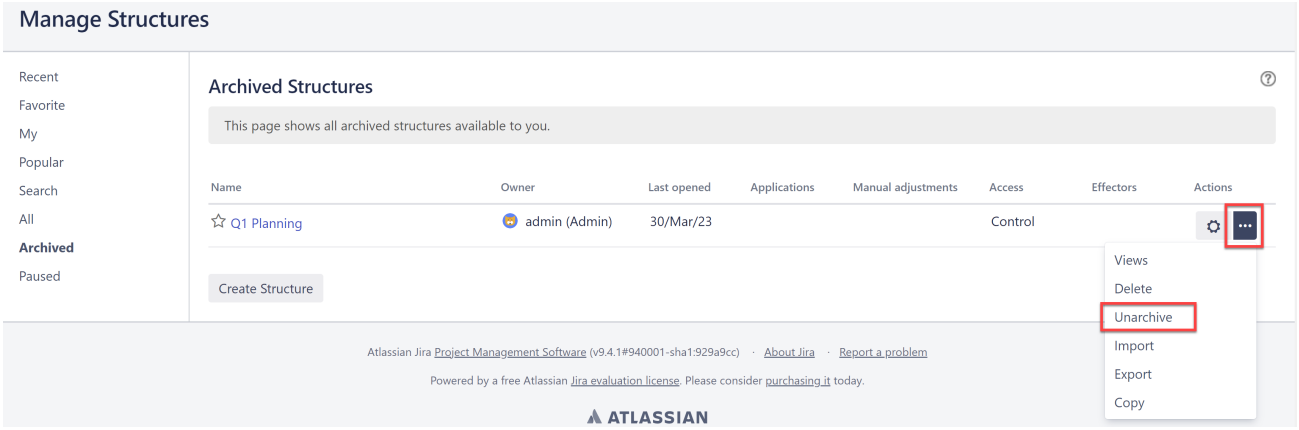
> You need **Control** access level to be able to archive a structure.

3.11.9.2 Unarchive a Structure

You can restore an archived structure to make it editable and visible in all menus.

To **unarchive** a structure:

1. Open the Manage Structures (see page 532)page using the top navigation **Structure** menu.
2. Select the **Archived** tab and locate the archived structure. You can also search for the structure using the **Search** tab – just remember to check the **Show Archived** box.
3. Once you locate the structure, open the Actions menu (...), and select **Unarchive**.



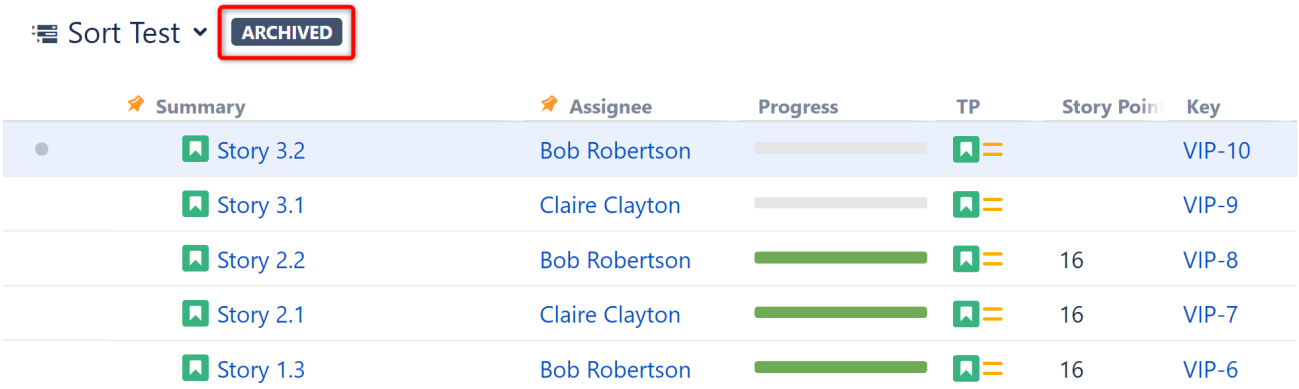
You need **Control** access level to be able to unarchive a structure.

3.11.9.3 Searching for Archived structures

Archived structures can be found on the following tabs on the **Manage Structures** page:

- **Archived** tab
- **Favorite** tab - if your favorites list contains any archived structures
- **Search** tab (Search by structure parameters) - you must check the **Show Archived** box
- **Search** tab (Search by the structure ID)

When an archived structure is opened, an **Archive** label appears next to the structure's name, so users can quickly identify it and know that it is read-only.



3.11.9.4 Generators (Automation) and Archiving

When you archive a structure built using generators(see page 140), the generators used to create the structure are saved, but not the issues they add. If you open or unarchive the structure at a later time, it may contain different issues and/or a different hierarchy from when it was archived, because the generators always build the structure based on current Jira values.

3.11.10 Deleting a Structure

Deleting a structure cannot be undone. If there is a chance you may need the structure in the future, consider archiving(see page 553) it instead.

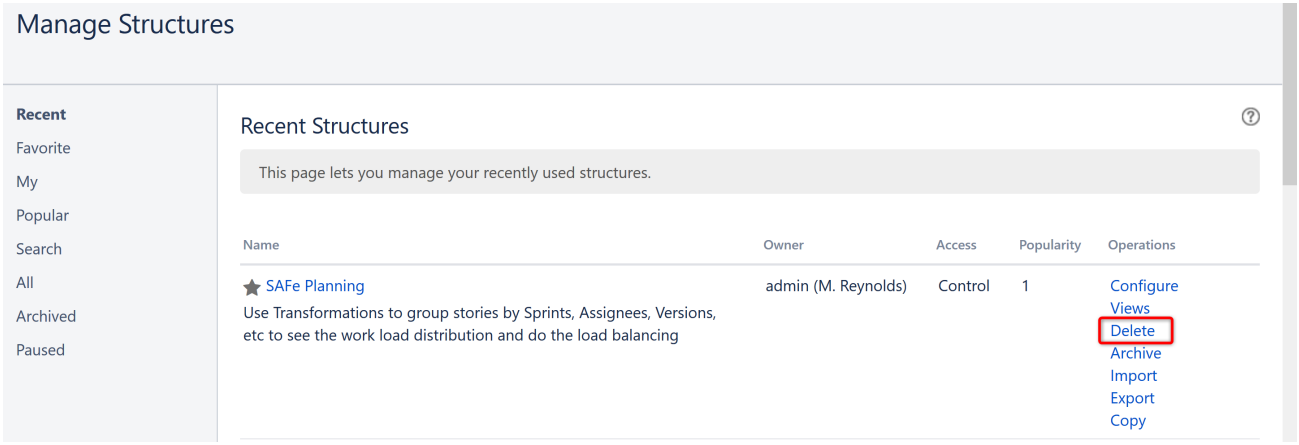
When you delete a structure, the following information is deleted:

- The hierarchical list of issues from the structure
- Structure details - name, description, permissions
- Automation rules for the structure
- Synchronizers installed into the structure

The issues within the structure are not affected in any way. They remain in Jira and can still be part of another structure.

To delete a structure:

1. Open the **Manage Structures** page using the top navigation **Structure** menu.
2. Find the structure you want to delete and click the **Delete** link in the Operations column.
3. Review the structure you are about to delete and confirm the operation. **Remember: this action cannot be undone!**



You need **Control** access level to be able to delete a structure. See Structure Permissions(see page 543) to learn more.

3.11.11 Template Structures and Projects

Template structure is a structure that you copy & clone to get the real, "workable" structures.

Technically, template structures are ordinary structures, containing ordinary issues. It is up to you to designate a structure to be a template and configure it accordingly.

3.11.11.1 Configuring Template Structures

Here are some suggestions about configuring template structures:

1. Clearly designate them as a template - for example, have "[Template]" marker as a part of the structure's name.
2. Give permissions to change the template structure only to those users who really need it. If needed, create another JIRA group for them (or ask JIRA administrator to do so).
3. Do not install any synchronizers on the template structure (unless you want the template to change, of course... which would be a quite unusual case).
4. Do not mark template issues as template in the issue summary. If you need to mark template issues somehow, use a label, which you will be able to remove from cloned templates via Bulk Change.

If you need to remove template issues from a JQL search, you can add to JQL: `AND NOT (issue in structure('template structure name'))`. See structure() JQL function(see page 428).

3.11.11.2 Creating Issues and a Structure from Template

Once you have a template structure, you can use Copy(see page 546) action from the **Manage Structures** page and turn on Clone Issues(see page 548) option. For details about configuring and running cloning operation, refer to Copying Structure and Cloning Issues(see page 548) article.

After you have created a new structure with new issues from template, you might want to:

- Rename the new structure and give it a meaningful name.
- Assign permissions for the new structure, if they are different from template structure permissions.
- Open the new structure to make sure it looks good.
- Do a Bulk Change(see page 138) on all issues - for example, to remove a template marker.

3.11.11.3 Template Projects

In the same manner, you can create a template project with template issues, and put them all into a template structure.

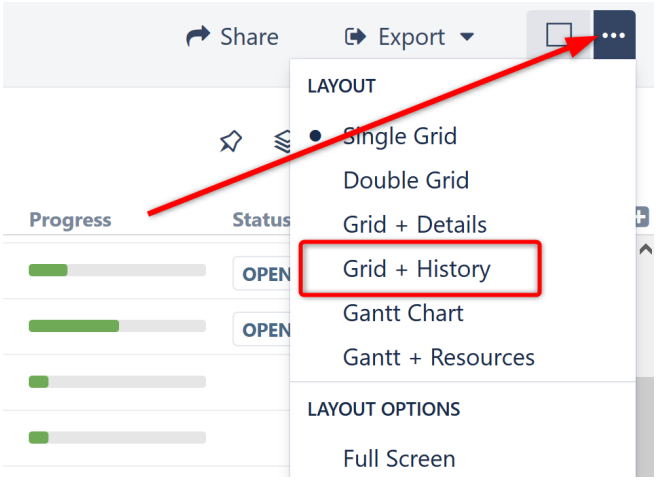
When you need to create a new project based on the template project, do the following:

1. Manually create an empty new project.
2. Create new structure and issues from template structure, as advised above. When configuring cloning parameters, specify the new project in the **Create in Project** parameter.

3.11.12 Viewing History of a Structure

Structure records every change that you or other users make to a structure. The History View lets you see those changes and previous versions of your structures.

To turn on History View, open the **Toggle Panels** menu and select **Grid + History**. The list of recorded changes will appear in the **History** panel on the right.



History does not work for dynamic parts of the structure. Changes made to issues added to the structure through Automation (see page 140)will not appear in the history. However, the addition, moving and removal of the generators themselves are recorded.

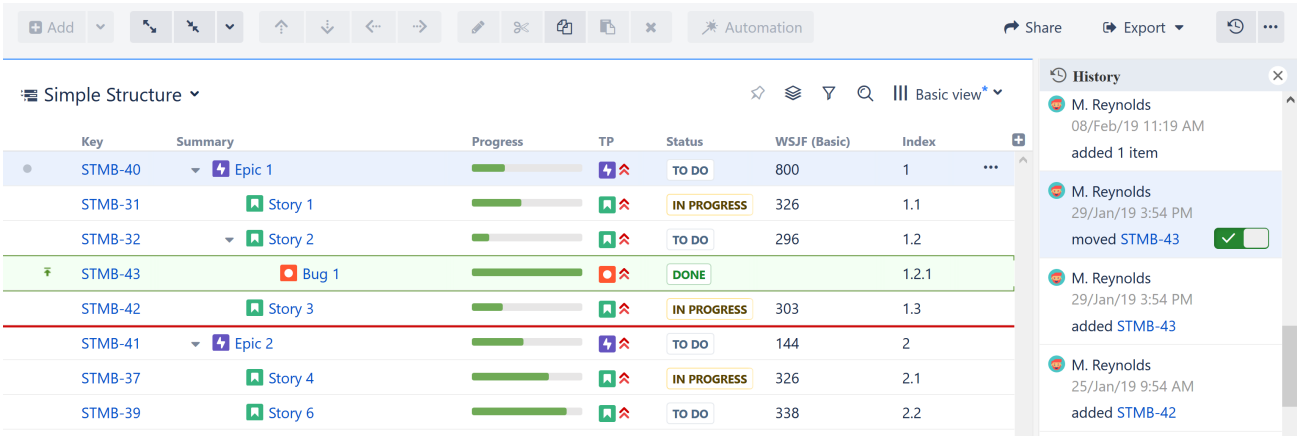
3.11.12.1 Reading History View

By default, the 20 most recent changes are loaded. To view additional changes, click the Show More button at the bottom of the list.

For each change, the following information is shown:

- The avatar and the name of the user who has made the change.
  - If the change was been made by a synchronizer, the synchronizer's name is shown. User avatar displays the user account that the synchronizer was running under.
- The nature of the change – how many issues were affected, were they added, removed or moved.
- The date and time when the change was made.

When you click a particular change, the main panel of the widget shows the structure as it was when that change was made. The affected issues are highlighted, and the structure expands and scrolls as needed to bring them into view.



Moved issues are shown in their new position by default, and their original position is marked by a red horizontal line. If you prefer to view moved issues in their original position, click the toggle button next to the entry in the history panel.

If issues were removed, they are shown in their position before the removal.

Use the **Ctrl+]** and **Ctrl+[** keyboard shortcuts to navigate to an earlier or later change.

3.11.12.2 Limitations of the History View

- History only tracks changes to the structure, not changes to Jira fields. Therefore, the values displayed in each column represents current Jira field values, **not** the values issues had when the structure change was made.
- History does not display changes that occurred through Generators(see page 140) - though it does display changes to the generators themselves.
- You cannot edit issues, create new issues or change structure when viewing history.
- The history cannot be modified. (The administrator can clear the entire Structure history.)

3.11.12.3 Printing a Previous Structure Version

You can create a printable version(see page 561) of the structure as it appeared following a given change. To do so, select the relevant change and click **Export | Printable Page**.

Please keep in mind that the printed structure will have the same limitations as the History View (Jira fields will display current values, etc.).

3.11.12.4 Exporting a Previous Structure Version to Excel

You can also export the structure to Excel(see page 558). To do so, select the relevant change and click **Export | Export to Excel**.

The XLS file will contain the structure as it appeared after the selected change was applied, with the same limitations as the History View (Jira fields will display current values, etc.).

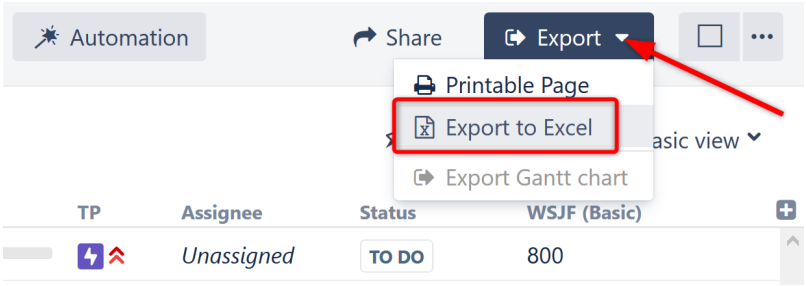
3.11.13 Exporting Structures

There are several options for exporting structures:

- Exporting to Excel(see page 558)
- Exporting to PDF(see page 561)
- Printing Structure(see page 561)

3.11.13.1 Exporting to Excel

You can download the structure that you see on the screen as an XLS file and open it in Microsoft Excel or in other applications that support this format.To export the structure to XLS, click the **Export** button in the toolbar and select **Export to Excel**.

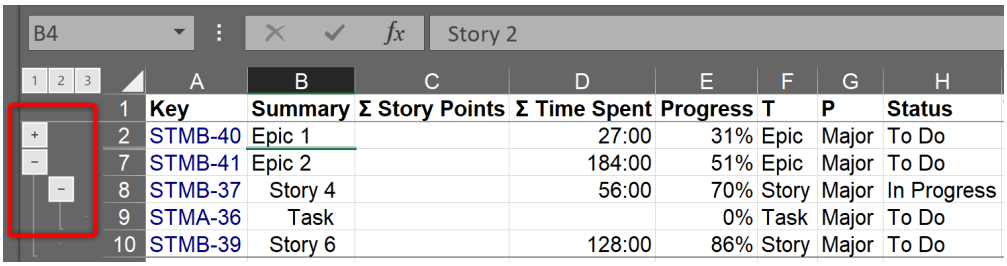


The browser will download a new XLS file, which you can save or open. The XLS file will:

- Contain all the issues that are present in the structure.
- Preserve the structure's hierarchy.
- Include the same columns as the structure.
- Display only the Summary field within the Summary column. To include the issue description, add a separate Description column to the structure before exporting.

Row Groups

The rows are grouped together using Excel's grouping feature to form a collapsible structure in the spreadsheet – you can expand and collapse sub-issues under their parent issues.



The maximum depth of grouping in an XLS file is 8. If you have a deeper structure, it will still be exported, but the grouping will only work for the top 8 levels.

> The structure hierarchy is also visualized using indentation in the Summary column, where up to 15 levels can be displayed.

Printing

The XLS file is set up for a standard printing configuration:

- Page orientation is Landscape.
- The content is fit horizontally on the page (you might need to change that if you have too many columns or large content).
- Paper size is set to *Letter* if your account locale is US or Canada; otherwise it is set to *A4*.

> It's a good idea to use Print Preview before sending the document for printing. If you don't like how it looks, consider using Printable page(see page 561).

Columns

The columns are formatted in the best way suitable for a spreadsheet.

Column Type	Notes
Issue Key	The cell with an issue key is a link to the actual issue.
Summary	Cells in the Summary column have indentation just like in Structure. Note that if you change the format of a cell, you might lose the indentation level.
Progress	Progress field contains a fractional number from 0 to 1, formatted as a percent value.
Description, Environment and large text fields	The text might not fit in the column. You can increase the column size or use the **Format Cells | Alignment | Wrap Text** option in Excel to have a large text take up more than one line, increasing the row height. Note that a cell might not accommodate a very large text, and you might only see the first part of it.
Dates	Date values are displayed in your local date format.
Estimates, Time Spent	Duration fields contain actual numbers (fractional number of days), which you can sum or otherwise process. The display format is **HH:MM**, where HH is the number of hours and MM is the number of minutes. So an estimation of 5 days will be displayed as **40:00** (if you have 8-hour work days). *Tip: If you prefer to see this in Jira Duration format (e.g., 1w 3d 5h 20m), try our* Excel Macro(see page 678).
Standard custom fields	Standard custom fields are rendered according to their type.
Custom fields from other add-ons	Custom fields from other add-ons are displayed as they are rendered.

Compatibility

The exported file is compatible with Microsoft Excel 2003+.

Technical Limitations

The XLS format allows a maximum of 65536 rows in a spreadsheet. If your structure contains more items than this, use filtering to hide some of the issues.

If your structure's hierarchy is more than 15 levels deep, any items deeper than level 15 will be indented to the 15th level in the Summary column of the XLS file.

Note for Add-on Developers

If your plugin provides a new custom field type, please ensure that the field is displayed with the best compatibility with the other plugins, including Structure. In your column view velocity template, check for `$displayParams.textOnly` and/or `$displayParams.excel_view` and/or `$displayParams.nolink` – all those parameters will be set to `true` by Structure and may also be used by other plugins. See `CommonVelocityKeys.java` and Jira sources for examples

3.11.13.2 Exporting to PDF

You can use one of the following methods to export a structure (or part of a structure) to PDF.

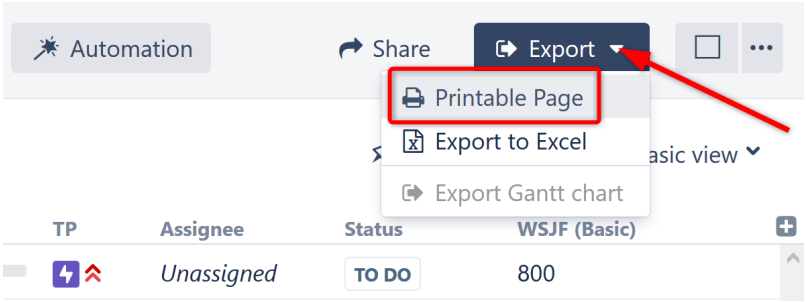
The built-in way: Print the structure (see page 561)and use your browser's built-in Print to PDF feature to turn it into a PDF.

The more convenient way: Use one of these partner add-ons to customize and even automate the export process. PDF export is not our specialty - it is for these amazing apps!

- Better PDF Exporter for Jira(see page 577)
- Xporter(see page 603)

3.11.13.3 Printing Structure

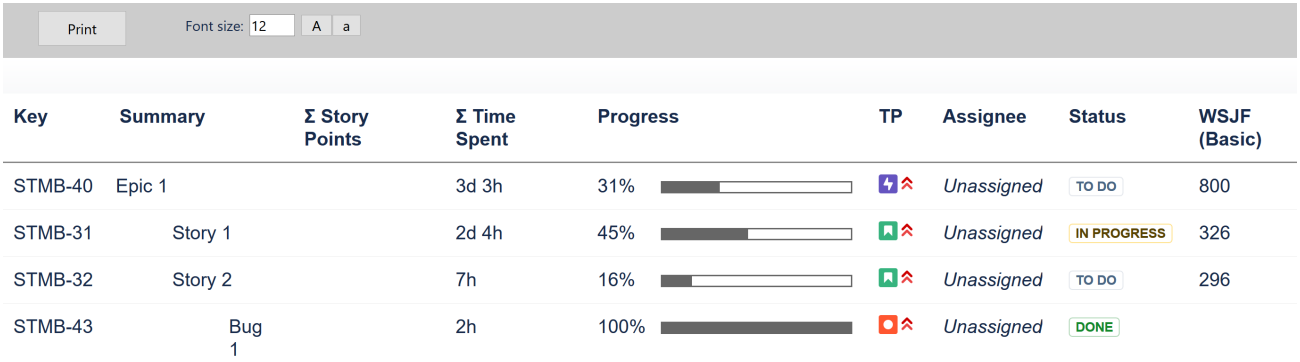
To print the current structure, click the **Export** button on the structure toolbar and select **Printable Page**.



The structure will open in a separate browser window or tab, with the following properties:

- It copies the current appearance of the structure, so if some sub-issues are hidden, you will not see them on the printable page.
- The columns displayed on the printable page will be the same as in the structure; however, column widths will be set by the browser.

- The Summary column on the printable page displays only the summary field, without the issue description or icons. If you'd like to print the descriptions, add a separate Description column to the structure.



Depending on the number of columns, and the amount of text, it may be necessary to adjust font size before printing. You can do so by entering a new font size or clicking the **A**/**a** buttons.

When you're ready to print, click the **Print** button or use your browser's Print menu.

It's a good idea to print a single sample page to make sure font size is big/small enough.

3.11.14 Real-Time Collaboration

Structure is a real-time collaboration tool. The hierarchy displayed in a structure is kept up-to-date with the Jira server, so:

- If someone else changes the structure on the server, you will see the web page update within several seconds.
- If someone edits an issue or otherwise changes it, the field values displayed within a structure will also be changed.
- If any part of a structure is built using Generators(see page 140), the inclusion and placement of issues within the structure will automatically update based on relevant changes in Jira.

Items that have been added, removed, moved or changed are highlighted for a second with a flashing yellow background.

Structure keeps data up-to-date by polling the server with short requests every few seconds when the application is in use. If Structure detects that the browser is inactive or that the user has switched to a different browser tab, it will reduce polling frequency or possibly stop polling altogether to conserve network traffic.

Polling should resume as soon as Structure detects activity, but if you ever feel that data may be out of sync, simply move the mouse or select a new issue.

3.11.15 Structure Activity Stream

JIRA's **Activity Stream** dashboard gadget lets you see recent activity in JIRA and other connected systems. The activity stream can be filtered (for example, by project) to show you only the changes that concern you or your team. In addition, **Activity** tab on the issue page displays recent activity that has affected the viewed issue.

With the Structure plugin installed, Activity Stream gadget may be configured to include changes made to structures. The activity stream on the issue page automatically includes all changes to all structures that affect the position of the viewed issue.

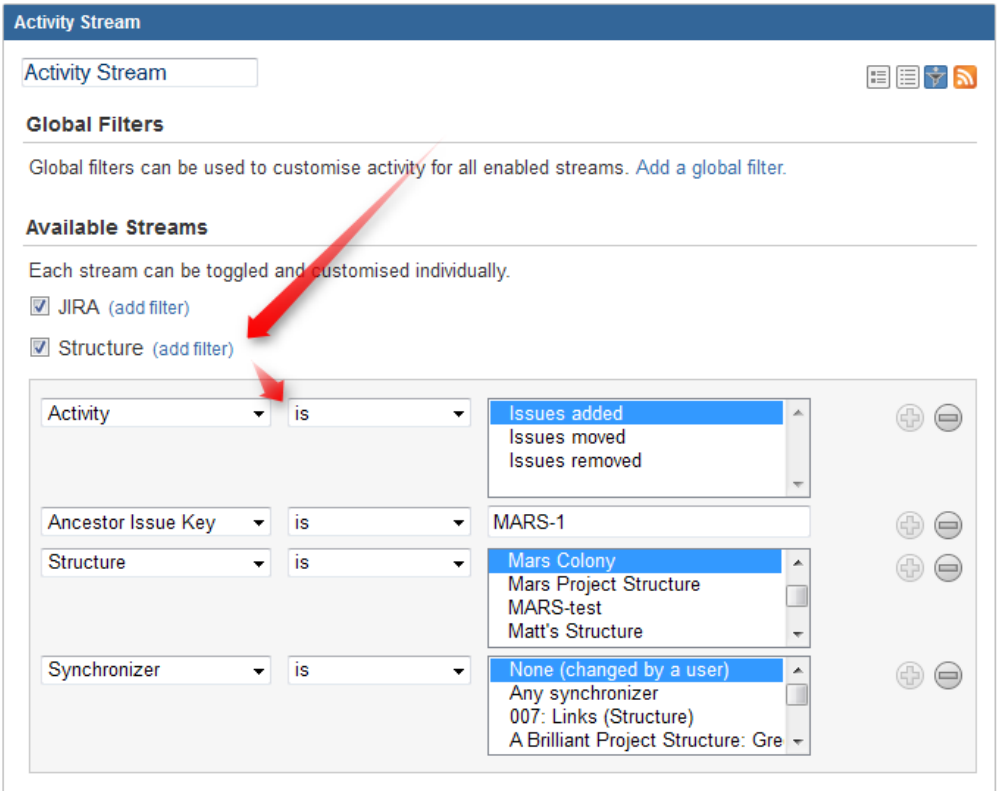
To activate the Structure stream, select the Structure option in the Available Streams section of the Activity Stream gadget configuration.

3.11.15.1 Available Filters

The following filters are available for the Structure activity stream:

- **Structure**
  Use it to see changes only in a specific structure or structures, or to exclude specific structures from the stream. If this filter is not used, changes to all structures are shown.
- **Ancestor Issue Key**
  This filter can be used together with the **Structure** filter if you are interested in changes within a specific part of a specific structure, located "under" the specified issue (if the changed issue is not located under the specified issue, the change will not be shown). You can enter several issue keys separated by spaces.
- **Synchronizer**
  You can include or exclude changes made by a synchronizer (either by any synchronizer or by specific synchronizers). Since synchronizers might make a lot of changes, this might be useful to filter out their "noise". Vice versa, you could verify that a synchronizer works as expected with an activity stream and this filter.
- **Activity**
  All changes to a structure fall into three categories: adding issues to structure, removing issues from structure and moving issues within structure. This filter lets you include or exclude the particular types of changes.

All Global Filters are supported by Structure Stream as well – you can filter structure changes by **Project, Issue Key, Update Date** and **Username**.



3.11.15.2 Reading Activity Stream

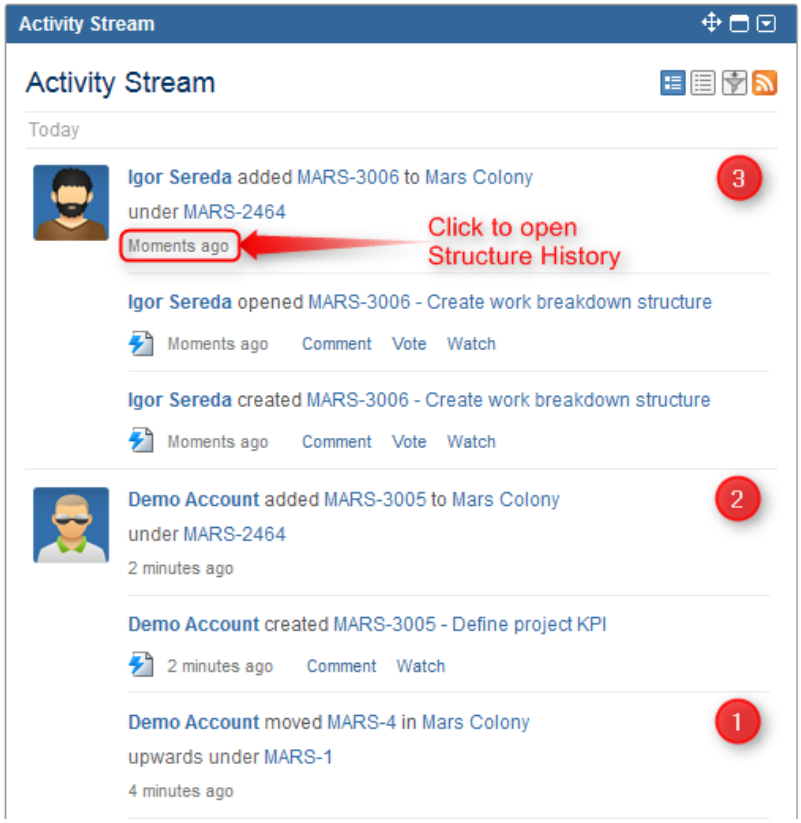
Changes in the Structure activity stream are ordered chronologically, newest first. For each change a short summary is displayed, containing:

- the full name of the user who made the change;
- for changes made by a synchronizer, the name of the synchronizer;
- the number of affected issues, and whether they were they added, removed or moved;
- if **Project** filter is used, the number of affected issues in each of the selected projects;
- if **Issue Key** filter is used, the affected issues among those selected in the filter;
- the name of the changed structure.

When viewing activity stream in the Full View, the following is also shown:

- the parent path of the affected issues;
- the original and the new parent path for the moved issues;
- if the issues were moved within the same parent issue, the direction of the move (upwards / downwards);
- when the change was made.

Parent Path is a sequence of issue keys: first, a top-level issue, then its sub-issue, then sub-sub-issue, and so on until the parent of the affected issue is displayed. Hover mouse over an issue key to view the issue's summary, or click it to go to that issue.



On this screenshot, items 1, 2 and 3 are Structure activities.

In the Full View, click on the time of the change to open that change on the Structure Board in the History View(see page 556).

3.11.15.3 Activity Streams Performance

Structure's activity stream is optimized to quickly provide data for the most common activity requests from Dashboard, Issue Activity, User Activity and Project Activity page.

It is possible however, if you use a complex search query on a JIRA instance with large history of structure changes, that querying database will take longer time than Activity Streams allows and you will not see any results. (There should be a message that "one of the activity streams providers took long time to provide an answer".)

If that is the case, try to reduce the amount of conditions you are using or contact support for help.

3.12 Structure Gadgets

Structure gadgets can be added to the Jira dashboard or Confluence pages.

See the following articles to learn more.

- Dashboard Gadget(see page 566)
- Confluence Gadget(see page 569)

3.12.1 Dashboard Gadget

You can view and edit structures directly from your Jira dashboard with the Structure dashboard gadget.

3.12.1.1 Adding Structure Gadget to Dashboard

Structure gadget can be added like any other dashboard gadget:

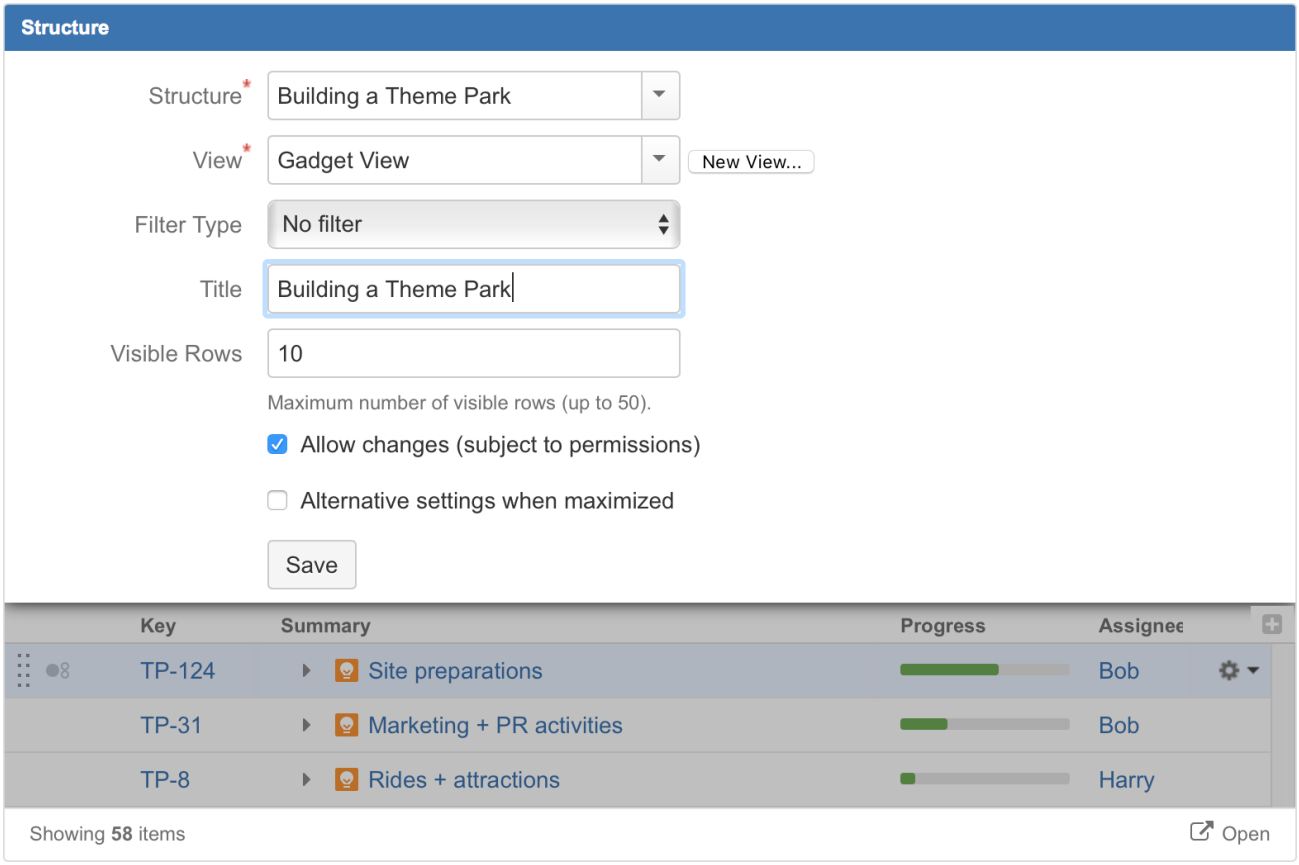
1. Click the **Add Gadget** button in the top right corner of the dashboard
2. Find "Structure" in the list of available gadgets (Hint: if you don't see Structure, click the **Load all gadgets** link)
3. Click **Add It Now**

In order to add a gadget to an existing dashboard, you must have change permissions. If you do not have change permission, you can try to create a copy of the dashboard using **Tools | Copy Dashboard**.

You can add several gadgets showing different structures on the same dashboard.

3.12.1.2 Configuring the Gadget

When you first add a gadget to dashboard, the gadget configuration panel appears with a dimmed preview of the gadget below. (The same panel is shown when you use the **Edit** command from the gadget header drop-down.)



To configure the gadget:

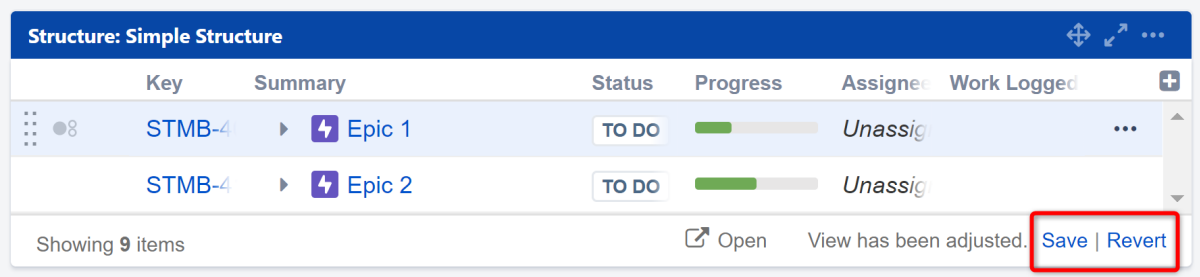
1. Select a **Structure**. Click arrow down in the Structure selector to view recently used and favorite structures, or start typing the structure name and let the drop-down suggest matching structures.
2. Select a **View**. Click arrow down to choose from views associated with the selected structure, start typing and let Structure suggest matching views, or create a new view. The selected view(see page 505) determines which columns the gadget displays. (You will be able to adjust the view later.)
3. Optionally, configure a **Filter**. The displayed structure will be filtered in the same way as in the Structure Board – see Filter(see page 207). You can choose between a text filter, JQL, S-JQL, or a saved JQL filter – see Search(see page 205).
4. Optionally, define the **Title** for this gadget. By default, it is the name of the selected structure.
5. Decide how large the gadget is allowed to be and specify the number of **Visible Rows**. If there are fewer visible rows, the gadget shrinks; if more, a vertical scroll bar appears. Pick any number between 2 and 50.
6. Decide whether dashboard viewers can make changes to the structure or issues (subject to the user's permissions) by selecting or deselecting the **Allow Changes** checkbox.
7. Optionally, if you would like the gadget to have a different **View** and **Visible Rows** settings when maximized, select the **Alternative settings when maximized** checkbox. This allows you to see more information for the same structure when the gadget window is maximized.
8. Click **Save**.

Deselect **Allow Changes** to protect the structure from accidental changes, such as changes caused by drag-and-drop or hitting the Delete key.

3.12.1.3 Customizing Gadget View

Once you have created your gadget, you can customize by adding, removing or rearranging columns – see Customizing Columns(see page 501).

Once you've made changes to the gadget's view, a new indicator will appear at the bottom of the gadget, along with options to save those changes or revert back to the view's original setting.



Save

Click the **Save** link will apply all the changes you have made to the existing view. The view will be updated everywhere it is in use, in the gadget, on Structure Board, etc. It will also impact all other users with access to that view.

You must have Update permission(see page 513) for the current view to save changes.

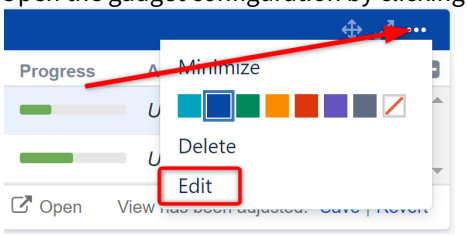
Revert

Click the **Revert** link to remove your adjustments and return to the original view as it is stored on the server.

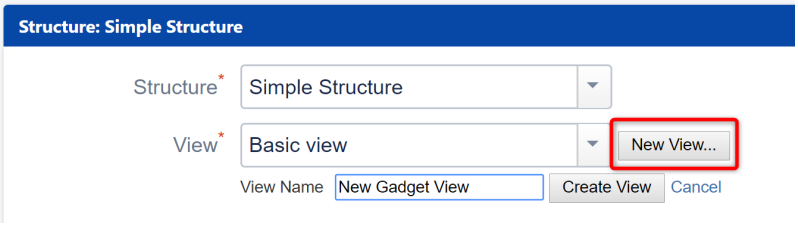
Save As

There is no Save As option on the dashboard gadget; however, it can be accomplished using the following method:

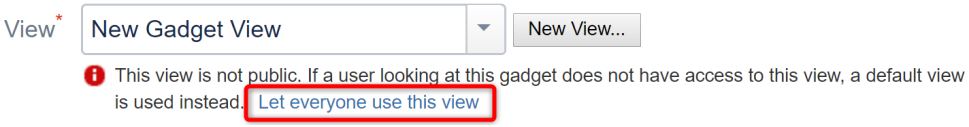
1. Adjust the gadget's view by adding, removing or rearranging columns – see Customizing Columns(see page 501) for details.
2. Open the gadget configuration by clicking **Edit** in the gadget header drop-down menu.



3. Click the **New View** button, located beside the view selector. An additional form appears – enter the new view name and click **Create View**.



4. If this gadget is going to be visible to other users, make sure they have access to the view you've created by clicking **Let everyone use this view**.



Use this method if you don't have **Update** access to the original view or want to continue using the original view elsewhere.

If the user viewing the gadget does not have **Use** permission on the configured view, the gadget will show a default view with only Issue Key and Summary as columns.

3.12.1.4 Using the Gadget

When viewing a structure in a gadget, some of Structure's features will be limited or unavailable:

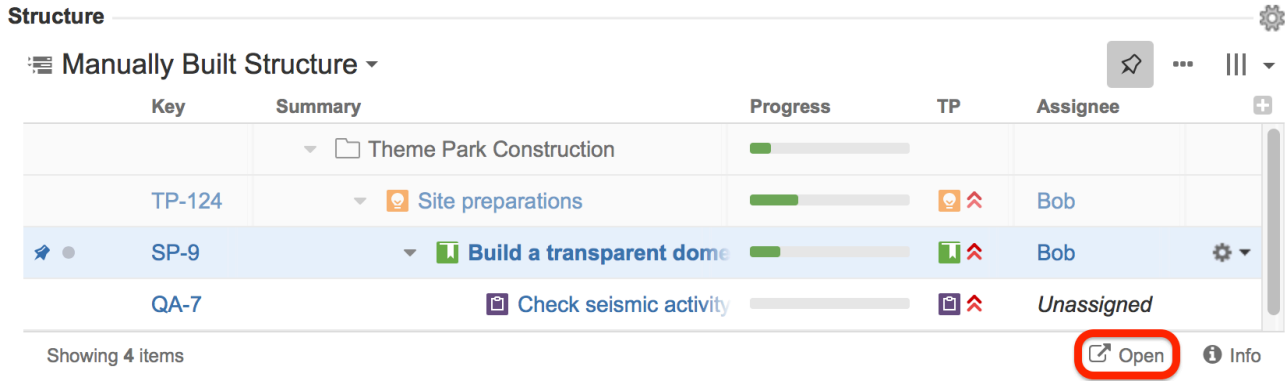
- Search and the Secondary panel is unavailable
- The toolbar is unavailable
- Only some fields can be edited from within Structure Gadget – see Editing from Gadget(see page 137)

However, many features are still available, including

- Most keyboard shortcuts are functional
- You can rearrange issues with drag-and-drop, if editing is enabled
- You can move, create, edit, and delete issues using the keyboard, if editing is enabled
- You can add, remove and rearrange columns
- If the gadget is displayed in its "home" Jira dashboard (not in Confluence or elsewhere), the last column lets you use the action drop-down for issues

3.12.1.5 Open on Structure Board

Working from the Structure Board(see page 88) provides the most unrestricted Structure experience. To get to the Structure Board from a gadget, click the **Open** link.

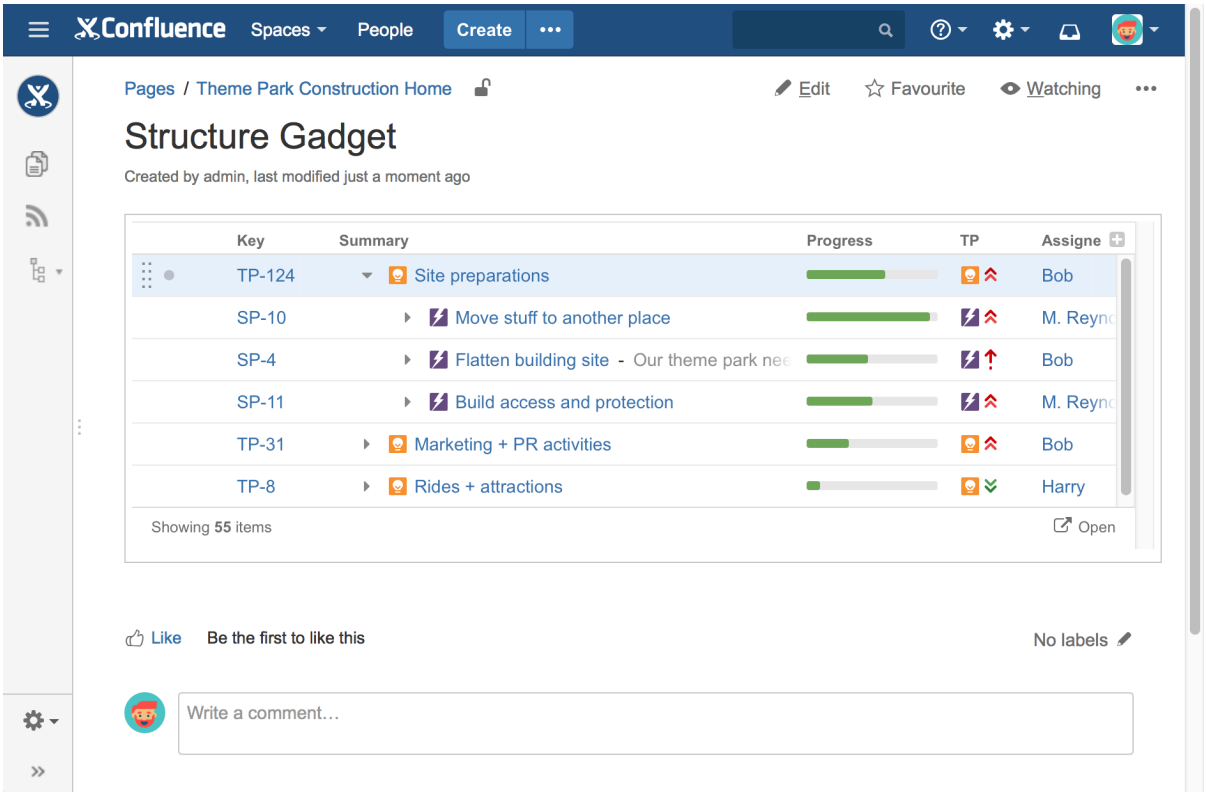


The structure will open on the Structure Board with the same filters[86] and transformations[87] that were applied in the original gadget. For example, if the gadget only shows items from a specific project, sorted by Assignee, that's exactly what you'll see on the Structure Board. To review or remove these transformations, click the Transformations button in the panel toolbar.

3.12.2 Confluence Gadget

You can embed a Structure gadget into a Confluence page to view the structure in Confluence.

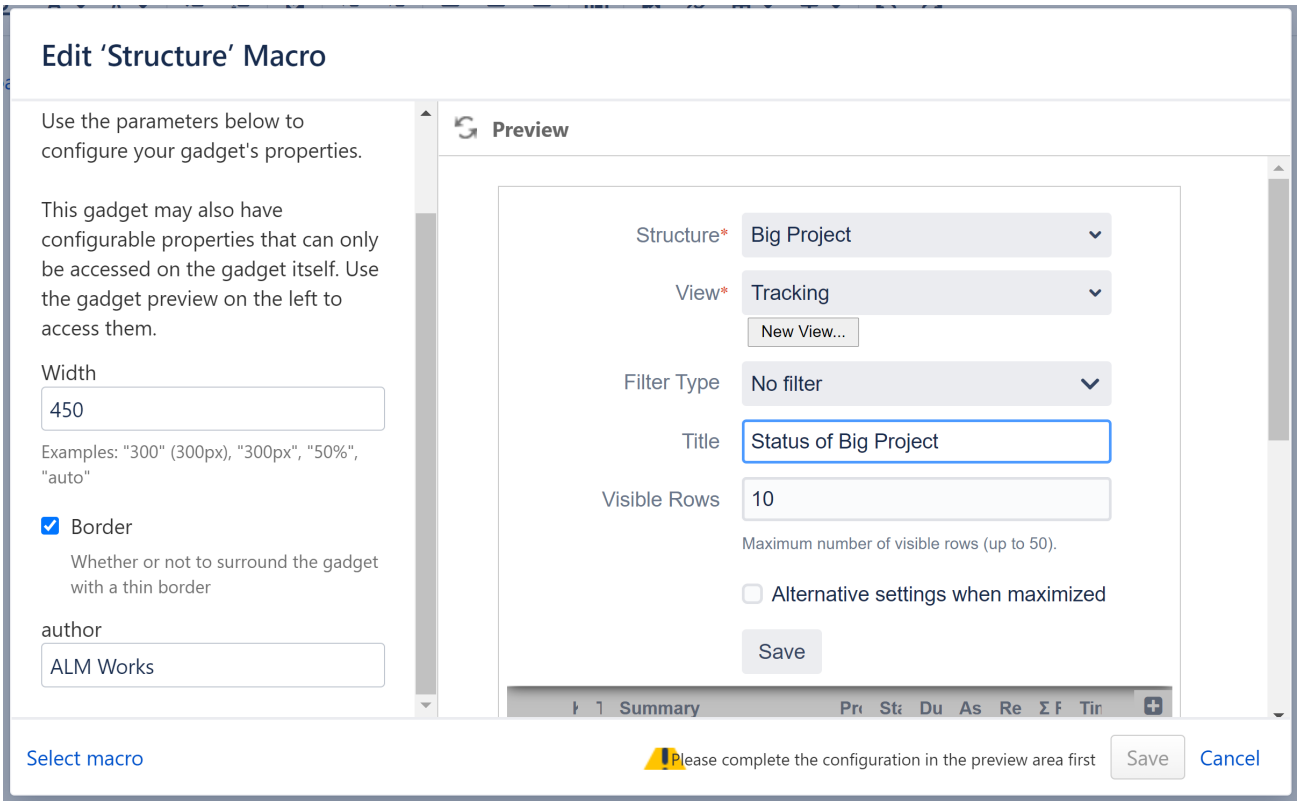
86 https://wiki.almworks.com/display/structure2gmaster/Filter
87 https://wiki.almworks.com/display/structure2gmaster/Transformations



Before you can use a Structure gadget on a Confluence page, your Confluence administrator must add Structure gadget to the Confluence configuration(see page 1010). If you try to insert a macro and don't see *Structure* in the available list, most likely the gadget is not configured.

3.12.2.1 Adding a Structure Gadget to a Confluence Page

When editing a Confluence page, click Insert/Edit Macro, and select **Structure**. The macro configuration dialog appears.



Select your gadget configuration:

1. Select a **Structure**. Click arrow down in the Structure selector to view recently used and favorite structures, or start typing the structure name and let the drop-down suggest matching structures.
2. Select a **View**. Click arrow down to choose from views associated with the selected structure, start typing and let Structure suggest matching views, or create a new view. The selected view(see page 505) determines which columns the gadget displays. (You will be able to adjust the view later.)
3. Optionally, configure a **Filter**. The displayed structure will be filtered in the same way as in the Structure Board – see Filter(see page 207). You can choose between a text filter, JQL, S-JQL, or a saved JQL filter – see Search(see page 205).
4. Optionally, define the **Title** for this gadget. By default, it is the name of the selected structure.
5. Decide how large the gadget is allowed to be and specify the number of **Visible Rows**. If there are fewer visible rows, the gadget shrinks; if more, a vertical scroll bar appears. Pick any number between 2 and 50.
6. Optionally, if you would like the gadget to have a different **View** and **Visible Rows** settings when maximized, select the **Alternative settings when maximized** checkbox. This allows you to see more information for the same structure when the gadget window is maximized.
7. Click **Save**.

Once you've configured the gadget, specify the appearance of the gadget on the left side of the screen:

- Specify the gadget's **width**
- Select whether or not it should have a **border**

Click **Insert** and you're done!

If you see the **Login & approve** button, you will need to log in to Jira before you can add the gadget.

A **Structure plugin not available** message indicates that you do not have any visible structure. This is most likely because you are not logged in.

3.12.2.2 Using the Gadget

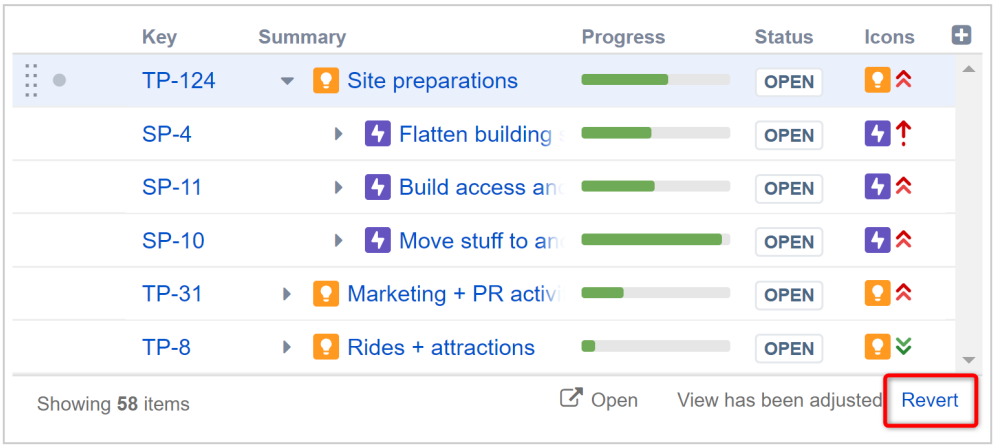
When viewing a structure in a gadget, some of Structure's features will be limited or unavailable:

- Search and the Secondary panel is unavailable
- The toolbar is unavailable
- The displayed Structure gadget is not suitable for printing. Please use Printable Page(see page 561) to print a structure separately.
- In the current version, it is not possible to edit the structure or issue fields from the Confluence gadget.

3.12.2.3 Customizing Gadget View

Once you have created your gadget, you can customize by adding, removing or rearranging columns – see Customizing Columns(see page 501).

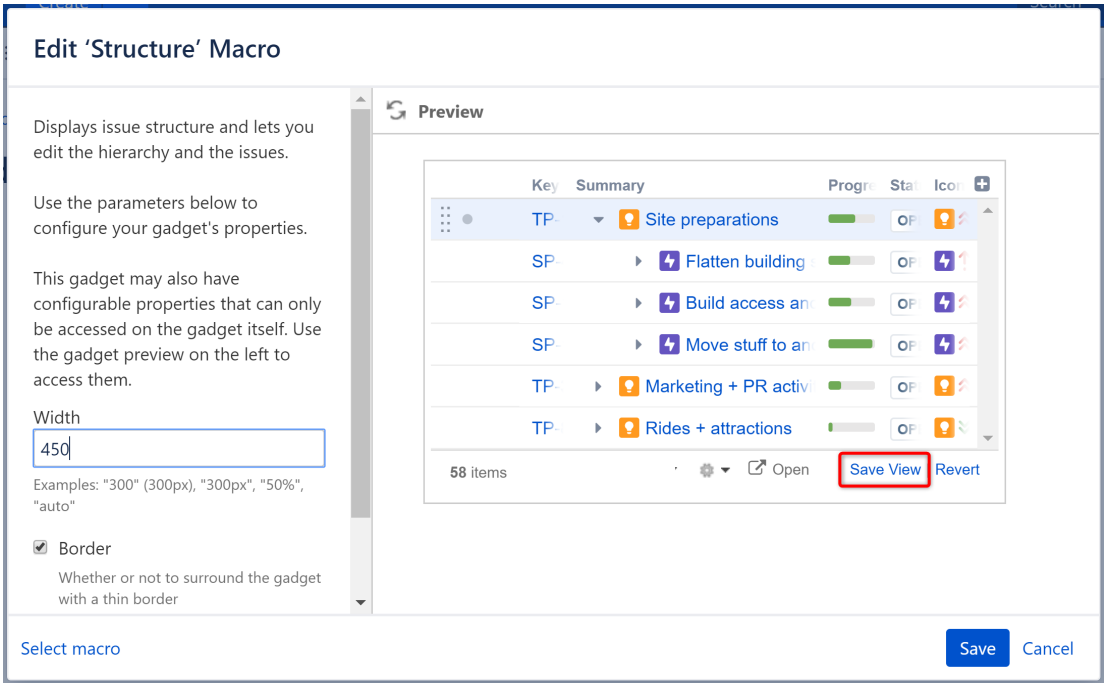
Once you've made changes to the gadget's view, a new indicator will appear at the bottom of the gadget, along with option revert back to the view's original setting.



In order to save the view, you need to enter Edit mode for the Confluence page and then edit the Structure macro. From there, you have a couple of options for saving your changes.

Save

Click the **Save** link to apply all the changes you have made to the existing view. The view will be updated everywhere it is in use, in the gadget, on Structure Board, etc. It will also impact all other users with access to that view.

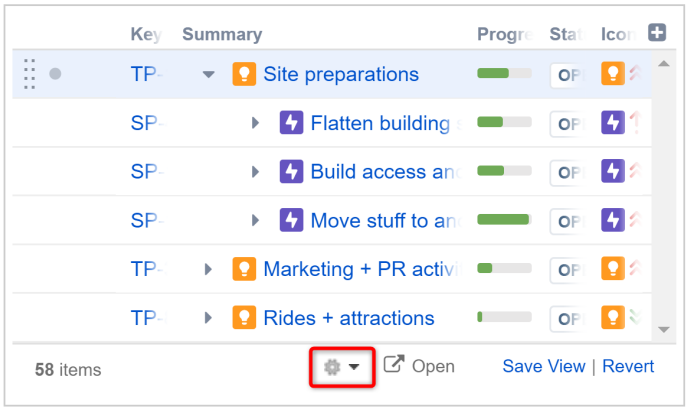


You must have Update permission(see page 513) for the current view to save changes.

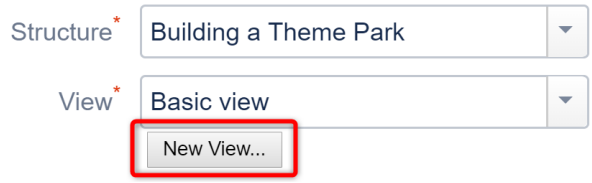
Save As

There is no Save As option on the Confluence gadget; however, it can be accomplished using the following method:

1. Adjust the gadget's view by adding, removing or rearranging columns – see Customizing Columns(see page 501) for details.
2. From the Edit 'Structure' Macro screen, open the gadget configuration by clicking the edit icon.



3. Click the **New View** button, located beside the view selector. An additional form appears – enter the new view name and click **Create View**.



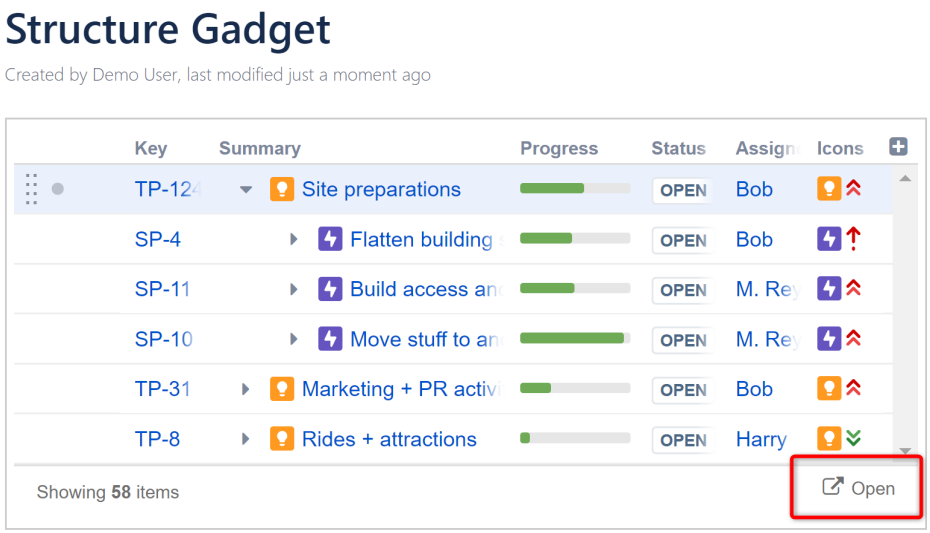
4. If this gadget is going to be visible to other users, make sure they have access to the view you've created by clicking **Let everyone use this view**.

Use this method if you don't have **Update** access to the original view or want to continue using the original view elsewhere.

If the user viewing the gadget does not have **Use** permission on the configured view, the gadget will show a default view with only Issue Key and Summary as columns.

3.12.2.4 Open on Structure Board

Working from the Structure Board(see page 88) provides the most unrestricted Structure experience. To get to the Structure Board from a gadget, click the **Open** link.



The structure will open on Structure Board with the same filters[88] and transformations[89] that were applied in the original gadget. For examples, if the gadget only shows items from a specific project, sorted by Assignee, that's exactly what you'll see on the Structure Board. To review or remove these transformations, click the Transformations button  in the panel toolbar.

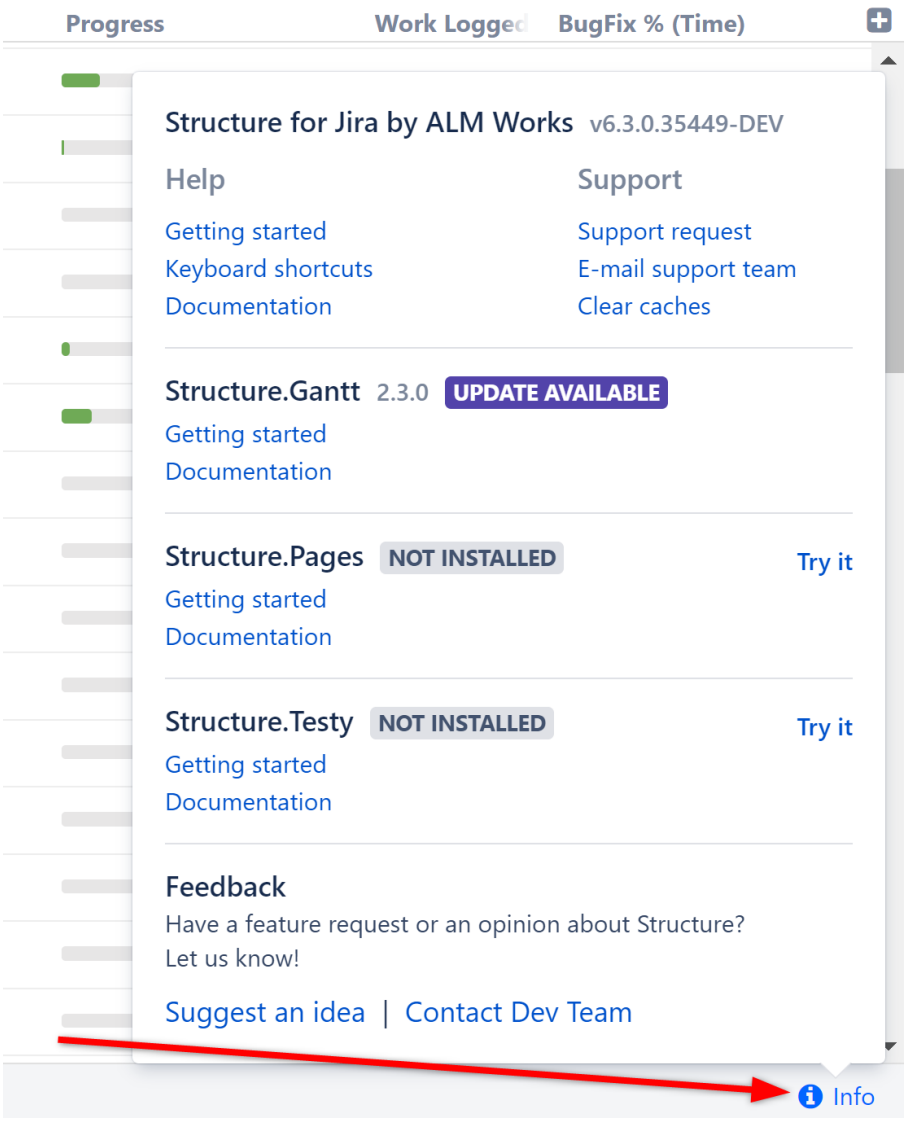
3.13 Getting Help

Should you have any questions or need assistance (whether regarding a feature of Structure or a personal need), we encourage you to refer to our Structure User's Guide[90] or contact our support team.

3.13.1 Resources Within Structure

If you have a question or need assistance while working with Structure, click the **Info** link at the bottom right corner of the Structure Widget. This contains links to documentation and keyboard shortcuts, version and update information for Structure and Structure extensions, and links for our support services.

88 https://wiki.almworks.com/display/structure2gmaster/Filter
89 https://wiki.almworks.com/display/structure2gmaster/Transformations
90 https://wiki.almworks.com/display/structure2gmaster/Structure+User%27s+Guide



3.13.2 Contact Us

Feel free to write back to ALM Works if you have any questions, feature requests or problems:

- Submit a support request, report a bug or suggest a feature[91]
- Suggest an idea[92] on our UserVoice forum
- Write to us[93] just to say hi or with any comments or questions

91 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
92 http://almworks.com/structure/suggest-idea
93 mailto:structure@almworks.com?subject=About%20Structure

3.14 Integrations

3.14.1 Advanced Roadmaps for Jira

Advanced Roadmaps for Jira[94] (formerly Portfolio for Jira) is a roadmapping solution for planning across teams and projects.

3.14.1.1 Key benefits to using Advanced Roadmaps with Structure

- Include Advanced Roadmaps fields as Structure columns
- Group issues by the Advanced Roadmaps Parent Link field - see Group Generators(see page 169)
- Extend issues using the Advanced Roadmaps Parent Link field - see Child Issues (Advanced Roadmaps) Extender(see page 155)

3.14.2 Better PDF Exporter for Jira

Better PDF Exporter for Jira[95] allows you to make customizable PDF exports from Jira. When installed with Structure, it allows you to create highly-customized PDF exports from any structure.

3.14.2.1 Key benefits to using Better PDF Exporter with Structure

- Fully customize your PDF export using their predefined templates or by designing your own export process
- Export automated structures to PDF - all generators are run prior to the export
- Support for Structure-specific items: folders, memos, and generators
- Build PDFs with embedded attachments, bookmarks, charts and more
- Export structures to time sheets, invoices, Release Notes, IEEE-standard Requirements Specification docs and more
- Automate the export process using Automation for Jira, ScriptRunner or RestAPI

Learn more: Better PDF Export for Jira - Structure Integration[96]

3.14.3 ScriptRunner

ScriptRunner is an app by Adaptavist[97] which allows the use of Groovy scripts to automate workflows, update fields and perform other actions in Jira. It also allows users to expand functionality of other apps–including Structure--using their APIs.

---

94 https://confluence.atlassian.com/advancedroadmapsserver
95 https://marketplace.atlassian.com/apps/5167/better-pdf-exporter-for-jira
96 https://www.midori-global.com/products/better-pdf-exporter-for-jira/server/documentation/integrations/structure
97 https://scriptrunner.adaptavist.com/latest/index.html

In the following pages, we have assembled some popular use-cases of Structure and ScriptRunner working together, along with links to the appropriate scripts in the Adaptavist Library[98].

3.14.3.1 Use-cases

- Creating (multiple) New Structures Programmatically(see page 582)
- Creating Generators with ScriptRunner(see page 583) - This script can be used to create new generators in a structure.
- Updating a field (ex. label) when checking all issues against a JQL query(see page 599) - This script will update a field (in this case Labels) for all issues that pass a certain JQL query.
- Automatically Remove Issues Based on JQL Query(see page 578) - This script will remove issues that do not satisfy a certain JQL query from a specific manually-built structure.
- Show All Structure Boards and Corresponding Item Counts(see page 594) - This script will show you the list of structures which exist in the instance and how many items were added to them manually. The result will be shown below the console.
- Bulk Change Owners of Structures and Generators(see page 581) - This script will bulk change the owner of structures from one user to another. This can be useful when one user account is being archived/removed and their structures need to be preserved.
- Show work logged per user and issue for a structure(see page 597) - This script will create a table showing users, each issue they logged work on and how many hours were logged for each particular issue.
- Run all effectors by schedule(see page 589) - This script will periodically run all effectors in the defined structure.
- Export to Excel Using ScriptRunner(see page 588) - This script will export a structure to excel.

3.14.3.2 Automatically Remove Issues Based on JQL Query

The following script will automatically review all issues in a manually built structure and remove any that do not satisfy the defined JQL query.

```
package examples.docs.structure

import com.atlassian.query.Query
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.row.StructureRow
import com.almworks.jira.structure.api.row.RowManager
import com.almworks.jira.structure.api.item.ItemIdentity
import com.almworks.jira.structure.api.item.CoreIdentities
```

98 https://library.adaptavist.com/collection/achieve-more-in-jira-with-scriptrunner-and-structure

```
import com.almworks.jira.structure.api.util.JiraComponents
import com.almworks.integers.LongArray
import com.almworks.integers.LongIterator
import com.almworks.integers.LongOpenHashSet

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version =
'16.10.0')

@WithPlugin("com.almworks.jira.structure")

@PluginModule
StructureComponents structureComponents

def plugin =
com.atlassian.jira.component.ComponentAccessor.pluginAccessor.getPlugin('com.alm
works.jira.structure')

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def permission = PermissionLevel.valueOf("ADMIN")



// Here we are going to get our structure and then get the forest that it is
built on
def structureName = "name1"
if (structureManager.getStructuresByName(structureName, permission).isEmpty()){
    log.warn "Something went wrong, couldn't find the structure."
 return
}
def struct = structureManager.getStructuresByName(structureName, permission)[0]
def forestSpec = ForestSpec.structure(struct.getId())
def forestSrc = forestService.getForestSource(forestSpec)
RowManager rowManager = structureComponents.getRowManager()
def forest = forestSrc.getLatest().getForest()

// This will allow us access to useful helper functions, in this case the
application of our JQL query to our rows.
```

```
def helper = plugin.getModuleDescriptor('helper').module


// This variable will hold all our issues that match our JQL query.
def matchingIssues = new LongOpenHashSet()
// This variable will store all the elements in our structure that are issues
(vs folders or generators).
LongArray onlyIssues = new LongArray()
// This variable will hold the rows that that are eventually removed from the
Structure.
LongArray matchingRows = new LongArray();


// This will turn our JQL string into a Jira query.
def jqlQuery = "assignee = Eve"
Query query =
JiraComponents.getComponent(com.atlassian.jira.jql.parser.JqlQueryParser).parseQ
uery(jqlQuery);



// Here we are iterating over all the rows of our structure to get the issues.
for (LongIterator ri : forest.getRows()) {
    StructureRow row = rowManager.getRow(ri.value())
    ItemIdentity itemId = row.getItemId()
    if (CoreIdentities.isIssue(itemId)) {
        onlyIssues.add(itemId.getLongId())
    }
}


// Here we are evaluating which items match the query. If we change the boolean
value to false, we get the opposite.
helper.matchIssues(onlyIssues, query, true, matchingIssues);


// Now we are iterating over our structure again, row by row, to translate the
issues that we want to remove to their respective rows.
for (LongIterator ri : forest.getRows()) {
    StructureRow row = rowManager.getRow(ri.value())
    ItemIdentity itemId = row.getItemId()
    if (CoreIdentities.isIssue(itemId) &&
matchingIssues.contains(itemId.getLongId())) {
```

```
        matchingRows.add(ri.value())
    }
}

// Here we pass the rows we identified as unwanted to our remove function.
forestSrc.apply(new
ForestAction.Remove(matchingRows.subList(0,matchingRows.size())))
```

3.14.3.3 Bulk Change Owners of Structures and Generators

```
import com.atlassian.jira.ComponentManager
import com.atlassian.jira.user.ApplicationUsers
import org.apache.log4j.Category

def oldOwnerKey = 'admin'
def newOwnerKey = 'ragnar'
def resync = true // do full resync after update

def plugin = ComponentManager.getInstance().getPluginAccessor().getPlugin('com.a
lmworks.jira.structure')
def loader = plugin.getClassLoader()
def StructureAuth = loader.loadClass('com.almworks.jira.structure.api.auth.Struc
tureAuth')
def JiraUser = loader.loadClass('com.almworks.jira.structure.api.permissions.Per
missionSubject$JiraUser')
def structureManager = plugin.getModuleDescriptor('structure-
manager').getModule()
def syncManager = plugin.getModuleDescriptor('sync-manager').getModule()

def oldOwner = JiraUser.newInstance(oldOwnerKey)
def newOwner = JiraUser.newInstance(newOwnerKey)
def newOwnerUser = ApplicationUsers.byKey(newOwnerKey)
if (newOwnerUser == null) {
    def message = "Cannot find user by user key: $newOwnerKey"
 log.error(message)
    return message
}

// The actual work is done here

def changedStructures = []
def changedSynchronizers = []
def success = false
def exception = null

try {
```

```
    StructureAuth.sudo {
        structureManager.getAllStructures(null, true).each { st ->
            // Change owner
 if (st.owner == oldOwner) {
                st.owner = newOwner
                st.saveChanges()
                changedStructures << st
            }
            // Change owner of synchronizers installed for this structure
 syncManager.getInstalledSynchronizersForStructure(st.id).each { sync ->
                if (sync.userKey == oldOwnerKey) {
                    def enabled = syncManager.isAutosyncEnabled(sync.instanceId)
                    if (enabled) {
                        syncManager.setAutosyncEnabled(sync.instanceId, false)
                        syncManager.updateSynchronizer(sync.instanceId,
sync.getParameters(), newOwnerUser)
                        if (resync) {
                            syncManager.resync(sync.instanceId, true, null)
                        } else {
                            syncManager.setAutosyncEnabled(sync.instanceId, true)
                        }
                    } else {
                        syncManager.updateSynchronizer(sync.instanceId,
sync.getParameters(), newOwnerUser)
                    }
                    changedSynchronizers << sync
                }
            }
        }
    }
    success = true
} catch (Exception e) {
    log.warn("Failed to change owner from '$oldOwnerKey' to '$newOwnerKey'", e)
    exception = e
}

// Output message about changed structures and synchronizers to the log and to
the console output
def msg = "Script to change owner from '$oldOwnerKey' for '$newOwnerKey' " +
        (success ? "finished successfully" : "failed (${exception &&
exception.message})") + "\n" +
        "Changed structures:\n" + changedStructures.collect({ "#${it.id} $
{it.name}" }).join("\n") + "\n" +
        "Changed synchronizers:\n" + changedSynchronizers.collect({ "#$
{it.instanceId} (for structure #${it.structureId})" }).join("\n")

log.warn(msg)
msg.replaceAll("\n", "<br>")
```

3.14.3.4 Creating a New Structure Programmatically

Running the following script you can create a new structure.

```
package examples.docs.structure

import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version =
'16.9.0')

@WithPlugin("com.almworks.jira.structure")

@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()

def permission = PermissionLevel.valueOf("ADMIN")

// It is important to check if an existing structure already exists; otherwise, this will
create a new structure with the same name.
if (structureManager.getStructuresByName("testScript", permission).isEmpty()){
log.warn "no existing Structure found"
// It is important to note that without the saveChanges() call at the end, the new
structure won't be saved to the database and will disappear after the script is run.
structureManager.createStructure().setName("testScript").saveChanges()
} else {
log.warn "found a pre-existing structure"
}
```

3.14.3.5 Creating Generators

Below are a series of scripts that run through the creation of an Insert, Extend, Sort and Filter Generator.

JQL Inserter

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.item.CoreIdentities
```

```
import com.almworks.jira.structure.api.generator.CoreGeneratorParameters
import java.util.Map

@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureName = "test" // name of the structure you want to add this
generator to
def jql = 'project = stmb and type = story' // this is the JQL query you want
the generator to executre

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def generatorManager = structureComponents.getGeneratorManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureId = structureManager.getStructuresByName(structureName,
permission)[0].getId()

// inserter
def params = new java.util.HashMap()
params.put(CoreGeneratorParameters.JQL, jql)
def jqlinserterId = generatorManager.createGenerator("com.almworks.jira.structur
e:inserter-jql", params, structureId)
def generatorItem = CoreIdentities.generator(jqlinserterId); // item to add to
forest
def forestSource =
forestService.getForestSource(ForestSpec.structure(structureId)) //resolving
forest source for structure
forestSource.apply(new ForestAction.Add(generatorItem, 0, 0, 0))
```

Extend stories under epics and extend by blocking link

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.item.ItemForestBuilderImpl;
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.generator.CoreStructureGenerators
import com.atlassian.jira.issue.link.IssueLinkTypeManager
import com.almworks.jira.structure.api.generator.CoreStructureGenerators
import java.util.Map
```

```
@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureName = "test"
def structureId = structureManager.getStructuresByName(structureName,
permission)[0].getId()
def forestBuilder = new ItemForestBuilderImpl()
def generatorManager = structureComponents.getGeneratorManager()

def epicsExtenderItem =
generatorManager.createGenerator(CoreStructureGenerators.EXTENDER_AGILE, [:],
null)
forestBuilder.nextRow(CoreIdentities.generator(epicsExtenderItem))

def issueLinkTypeManager = ComponentAccessor.getComponent(IssueLinkTypeManager)
def blocksLinkTypes = issueLinkTypeManager.getIssueLinkTypesByName('Blocks')
def blocksLinkTypeId = blocksLinkTypes[0].id

def params = [
        'linkTypeId': blocksLinkTypeId, // id of the Blocks issue link type
 'direction': 'outward',         // direction of the link type, could be either
'inward' or 'outward'
 'disableActions': false,        // enable structure actions, i.e. issues DnD
 'from': 2,
        'to': 2 // number of levels to extend issue at
]
def blocksExtenderItem =
generatorManager.createGenerator(CoreStructureGenerators.EXTENDER_LINKS, params,
structureId)
forestBuilder.nextRow(CoreIdentities.generator(blocksExtenderItem))

def forestService = structureComponents.getForestService()
def forestSource =
forestService.getForestSource(ForestSpec.structure(structureId))
def forestToAdd = forestBuilder.build()
forestSource.apply(new ForestAction.Add(forestToAdd, 0, 0, 0))
```

Sorting by a formula result and manual sorter

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
```

```
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.item.ItemForestBuilderImpl;
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.generator.CoreStructureGenerators

@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureName = "test"
def structureId = structureManager.getStructuresByName(structureName,
permission)[0].getId()
def forestBuilder = new ItemForestBuilderImpl()
def generatorManager = structureComponents.getGeneratorManager()

def manualSorterItem =
generatorManager.createGenerator(CoreStructureGenerators.SORTER_MANUAL, [:],
null)
forestBuilder.nextRow(CoreIdentities.generator(manualSorterItem))
/* adding formula-based sorter
*/
def formulaSorterParams = [
        attribute: [
                id : "expr",
                format: "order",
                params : [
                        formula: "assignee + key",
                        variables: [
                                assignee: [ id: "Assignee", format: "text"],
                                key: [ id: "Key", format: "text"]
                        ]
                ]
        ]
]
def formulaSorterItem =
generatorManager.createGenerator(CoreStructureGenerators.SORTER_ATTRIBUTE,
formulaSorterParams as Map, null)
forestBuilder.nextRow(CoreIdentities.generator(formulaSorterItem))
def forestService = structureComponents.getForestService()
def forestSource =
forestService.getForestSource(ForestSpec.structure(structureId))
def forestToAdd = forestBuilder.build()
forestSource.apply(new ForestAction.Add(forestToAdd, 0, 0, 0))
```

Below adds a JQL filter to a structure

```
package examples.docs.structure

// Structure imports
```

```
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.generator.CoreGeneratorParameters
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents

// Scriptrunner imports
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin

// Atlassian import (might be available without the import in some instances,
but better safe than sorry)
import com.atlassian.jira.component.ComponentAccessor

// A Hashmap
import java.util.Map

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version =
'16.9.0')

@WithPlugin("com.almworks.jira.structure")

@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def generatorManager = structureComponents.getGeneratorManager()

// For brevity's sake we will have a permission variable that we pass when we
make an admin level request
def permission = PermissionLevel.valueOf("ADMIN")

// the false flag is optional, it is a variable whether we want to search
archived structures too
// we are getting the first (zeroth) element because this method returns a list,
be mindful if you have multiple structures with the same name!
def currentStructure = structureManager.getStructuresByName("testScript",
permission, false)[0]


def jqltext = "assignee=admin";
Map<String, Object> jqlparams = new java.util.HashMap()
jqlparams.put(CoreGeneratorParameters.JQL, jqltext)

// We have now added our JQL text to the filter (creating a different kind of
inserter works almost exactly the same way, just build up the params variable
differently)
def jqlfilterId = generatorManager.createGenerator("com.almworks.jira.structure:
filter-jql", jqlparams, currentStructure.getId())

def generatorItem = CoreIdentities.generator(jqlfilterId); // item to add to
```

```
forest
def forestSource =
forestService.getForestSource(ForestSpec.structure(currentStructure.getId())) //
resolving forest source for structure
forestSource.apply(new ForestAction.Add(generatorItem, 0, 0, 0))
```

3.14.3.6 Export to Excel Using ScriptRunner

This script can be used in ScriptRunner[99] to run export. Use ScriptRunner Jobs[100] to add scheduling.

Export to excel

```
import com.atlassian.jira.component.ComponentAccessor
String fileName = 'filename.xls'
String userName = 'admin'
Long structureId = 1L
Long viewId = 2L
String viewSpec = null
String expandState = null
String clientFilter = null
def plugin = ComponentAccessor.pluginAccessor.getPlugin('com.almworks.jira
.structure')
def helper = plugin.getModuleDescriptor('helper').module
def ExcelExporter = plugin.classLoader.loadClass('com.almworks.jira.struct
ure.export.excel.ExcelExporter')
def exporter = helper.instantiate(ExcelExporter)
def ForestSpec = plugin.classLoader.loadClass('com.almworks.jira.structure
.api.forest.ForestSpec')
def spec = ForestSpec.structure(structureId,
helper.userManager.getUser(userName))
exporter.configure(viewId, viewSpec, spec, expandState, clientFilter)
exporter.prepareExport()
exporter.createWorkbook({null})
def stream = new FileOutputStream(fileName)
try {
    exporter.writeWorkbookTo(stream)
} finally {
    stream.close()
    return new File(fileName).toPath().toAbsolutePath()
}
```

99 https://scriptrunner.adaptavist.com/
100 https://scriptrunner.adaptavist.com/latest/bamboo/ScriptJobs.html

3.14.3.7 Run all effectors by schedule

This script can be used in ScriptRunner[101] to run effectors by schedule. See ScriptRunner Jobs[102].

101 https://scriptrunner.adaptavist.com/
102 https://scriptrunner.adaptavist.com/latest/bamboo/ScriptJobs.html

Run effectors

```
/*
 * This script allows to start specified or all effectors for chosen structure.
 * It will ping started effector process until it finishes.
 * It will NOT acknowledge the process at the end so the process will be available in the structure status bar process list.
 */

// params
def structureId = 14 // target structure ID
def effectorIds = [2, 1] // effector IDs to run or empty to run all effectors in structure
def processCheckInterval = 1000; // milliseconds

// Structure imports
import com.almworks.jira.structure.api.StructurePluginHelper
import com.almworks.jira.structure.api.auth.StructureAuth
import com.almworks.jira.structure.api.effector.instance.EffectorInstanceManager
import com.almworks.jira.structure.api.effector.process.EffectorProcessManager
import com.almworks.jira.structure.api.error.StructureException
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.ForestService
import com.almworks.jira.structure.api.forest.ForestSource
import com.almworks.jira.structure.api.StructureComponents
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.process.ProcessHandleManager
import com.almworks.jira.structure.api.process.ProcessStatus
import com.almworks.jira.structure.api.row.RowManager
import com.almworks.jira.structure.api.structure.StructureManager
import com.almworks.jira.structure.api.util.StructureUtil

// ScriptRunner imports
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.onresolve.scriptrunner.runner.ScriptRunnerImpl
// Atlassian import (might be available without the import in some instances, but better safe than sorry)
import com.atlassian.jira.component.ComponentAccessor

import static java.util.Arrays.asList

// switch to plugin
@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version = '*')
@WithPlugin('com.almworks.jira.structure')

```

```
43  class EffectorRunner {
44    StructureComponents SC =
    ScriptRunnerImpl.getPluginComponent(StructureComponents)
45    ForestService FS = SC.getForestService()
46    RowManager RM = SC.getRowManager()
47    StructureManager SM = SC.getStructureManager()
48    ProcessHandleManager PHM = SC.getProcessHandleManager()
49    // next components are not available from StructureComponents in
    Structure 6.x, but who knows :)
50    EffectorInstanceManager EIM =
    ScriptRunnerImpl.getPluginComponent(EffectorInstanceManager)
51    EffectorProcessManager EPM =
    ScriptRunnerImpl.getPluginComponent(EffectorProcessManager)
52
53    def user = StructureAuth.getUser()
54    ForestSource source
55
56    def structureId
57    def effectorIds
58    def log
59    def processCheckInterval
60
61    EffectorRunner(def log, def structureId, def effectorIds, def
     processCheckInterval) {
62      this.structureId = structureId
63      this.effectorIds = effectorIds
64      this.log = log
65      this.processCheckInterval = processCheckInterval
66      try {
67        // check for structure existence and permissions
68        SM.getStructure(structureId, PermissionLevel.VIEW)
69        source = FS.getForestSource(ForestSpec.skeleton(structureId))
70      } catch (StructureException ex) {
71        log.error("Cant create forest source for structure id=${structureId}
    for user ${user}.\n${trimStackTrace(ex)}")
72      }
73    }
74
75    def run() {
76      def caption = "See logs for results."
77      if (source == null) return caption
78      def structureName =
    FS.getDisplayName(ForestSpec.skeleton(structureId))
79
80      def effectors = getEffectorInstances()
81      if (effectors.isEmpty()) {
82        log.warn("No effectors were found.")
83        return caption
84      }
85
86      if (!effectorIds.isEmpty() && effectorIds.size != effectors.size()) {
87        return caption
88      }
```

```
89
90      log.warn("Starting to execute effectors for structure '${structureName}' for user ${user}")
91      // print effectors list to run
92      effectors.each {ef ->
93        log.warn("\teffector: id=${ef.getId()} ${ef.getModuleKey()} ${ef.getParameters()}")
94      }
95
96
97
98
99      // start effectors process
100     def processId = runEffectors(effectors)
101     if (processId == null) return caption
102
103     log.warn("Effector process id=${processId} started.")
104
105     // await process finish
106     if (awaitFinish(processId)) {
107       // print produced effect records and errors
108       printResults(processId)
109       log.warn("Done.")
110     } else {
111       log.error("Failed.")
112     }
113
114     return caption
115   }
116
117   def printResults(def processId) {
118     def records = EPM.getEffectRecords(processId)
119     if (records.isEmpty()) {
120       log.warn("No effect records were produced.")
121     } else if (records.size() > 100) { // just not to spam
122       log.warn("There are ${records.size()} applied effects.")
123     } else {
124       records.each {record ->
125         // choose right log level for errors and applied effects
126         if (record.isError()) {
127
log.error(StructureUtil.getTextInCurrentUserLocale(record.getEffectMessage()))
128         } else {
129
log.warn(StructureUtil.getTextInCurrentUserLocale(record.getEffectMessage()))
130         }
131       }
132     }
133
134   }
135
```

```
136   // This is blocking method will run until effectors process finish.
137   def awaitFinish(def processId) {
138     def handleId
139     try {
140       def process = EPM.getProcess(processId)
141       handleId = process.getProcessHandleId()
142     } catch (StructureException ex) {
143       log.error("Failed to get process info\n.${trimStackTrace(ex)}")
144       return false
145     }
146
147     def info = PHM.getInfo(handleId)
148     if (info == null) return false
149
150     def prevProgress = 0
151     while (info != null && info.getStatus() != ProcessStatus.FINISHED &&
      sleep()) {
152       info = PHM.getInfo(handleId)
153       if (info.percentComplete != prevProgress) {
154         def process = EPM.getProcess(processId)
155         log.warn("\tprocess complete ${info.getPercentComplete()}%. Status
      ${process.getStatus()}")
156         prevProgress = info.getPercentComplete()
157       }
158     }
159     return info != null
160   }
161
162   def sleep() {
163     try {
164       Thread.sleep(processCheckInterval)
165       return true
166     } catch (InterruptedException ex) {
167       log.warn(${trimStackTrace(ex)})
168       Thread.currentThread().interrupt()
169       return false
170     }
171     return true
172   }
173
174   def runEffectors(def effectors) {
175     try {
176       return EPM.startProcess(effectors, structureId, false)
177     } catch (StructureException ex) {
178       log.error("Effector process validation failed.\n$
      {trimStackTrace(ex)}")
179       return null
180     }
181   }
182
183   def getEffectorInstances() {
184     def ids = effectorIds.isEmpty() ? findEffectors() : effectorIds
185     def instances = []
```

```
186       ids.each {id ->
187         try {
188           instances.add(EIM.getEffectorInstance(id))
189         } catch (StructureException ex) {
190           log.error("Unable to find effector with id=${id}.\n$
    {trimStackTrace(ex)}")
191         }
192       }
193
194       return instances
195     }
196
197     def findEffectors() {
198       def effectorIds = []
199       source.getLatest().getForest().forEach {row ->
200         def rowId = row.left()
201         def itemId = RM.getRow(rowId).getItemId()
202         if (CoreIdentities.isEffector(itemId)) {
203           effectorIds.add(itemId.getLongId())
204         }
205       }
206       return effectorIds
207     }
208
209     def trimStackTrace(Exception e) {
210       return asList(e.stackTrace).subList(0, 20).join('\n    ') +
    '\n    ...'
211     }
212   }
213
214   new EffectorRunner(log, structureId, effectorIds,
    processCheckInterval).run()
```

3.14.3.8 Show All Structure Boards and Corresponding Item Counts

This script will show you the list of structures on the instance and the number of items added to each structure manually. These manual items include Generators.

```
import com.almworks.jira.structure.api.StructureComponents
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.util.Util
import com.atlassian.jira.component.ComponentAccessor
import com.onresolve.scriptrunner.runner.customisers.WithPlugin

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version =
'16.9.0')
@WithPlugin('com.almworks.jira.structure')

StructureComponents structureComponents =
ComponentAccessor.getOSGiComponentInstanceOfType(StructureComponents)
```

```
def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()

def structures = structureManager.getAllStructures(PermissionLevel.NONE)
def map = structures.collectEntries {
  def fs = forestService.getForestSource(ForestSpec.skeleton(it.id))
  [(it.name + ' (#' + it.id + ')'): fs.getLatest().getForest().size()]
}

map = map.sort {-it.value}

'<table><tr><th>Structure Name (ID)</th><th>Number of manually added rows</th></
tr>' +
map.collect {'<tr><td>' + Util.htmlEncode(it.key as String) + '</td><td>' +
it.value + '</td></tr>'}.join('')  +
  '</table>'
```

This script can be adopted to show the count of dynamically generated items by replacing ***ForestSpec.skeleton()*** with ***ForestSpec.structure()***. However, running this script will trigger all the structures and their generators, which can cause a performance impact. Be careful when running this in a production environment.

3.14.3.9 Show Related Issues in a Separate Column

The following two scripts can be used to work with issues on two separate instances connected with an App Link. It adds a field to the issue that shows all the issues that are linked through any issue link type. The first script does this purely for issues that are on a linked, but separate, Jira instance (like a Service Management / Service Desk instance and a Dev instance, for example). The second script does the exact same thing for all linked issues on the same instance.

In order to run these scripts, there are a few preconditions that need to be met. You will need to create a scripted field[103] and set it to an HTML template.

Script 1

This script identifies issues on a separate Jira instance that have an issue link connecting them to any issues on this instance.

Be aware that in order for this script to work, these instances will need to be connected through an Application Link and the user running these scripts will need Jira administrator privileges on both instances.

```
package com.onresolve.jira.groovy.test.scriptfields.scripts

import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.link.RemoteIssueLinkManager
```

103 https://scriptrunner.adaptavist.com/latest/jira/scripted-fields.html

```
def remoteLinkManager = ComponentAccessor.getComponent(RemoteIssueLinkManager)

def startHTML = '<a href="'
def endHTML = '">'

def remoteIs = remoteLinkManager.getRemoteIssueLinksForIssue(issue)
def remIs = new String[remoteIs.size()]
def count = 0
remoteIs.each {
    remIs[count] = startHTML + it.getUrl() + endHTML + it.getTitle() + '</a>'
    count++
}
remIs
?:null
```

Script 2

This script will add a field to each issue that shows all the linked issues on this same instance.

```
package com.onresolve.jira.groovy.test.scriptfields.scripts

import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.link.IssueLinkManager

def linkManager = ComponentAccessor.getComponent(IssueLinkManager)

def startHTML = '<a href="'
def baseUrl = 'http://myjirainstance.com/browse/'
def endHTML = '">'

def issueLinks = linkManager.getIssueLinks(issue.id)
def remIs = new String[remoteIs.size()]
def count = 0
def otherIssue
issueLinks.each {
    if( issue.id == it.id ){
        otherIssue = it.getDestinationObject()
    } else {
        otherIssue = it.getSourceObject()
    }
    remIs[count] = startHTML + baseUrl + otherIssue.getKey() + endHTML +
otherIssue.getKey() + '</a>'
    count++
}
remIs
?:null
```

⚠ You should replace the value in `baseUrl` with the actual URL of your local instance.

3.14.3.10 Show work logged per user and issue for a structure

```
package examples.docs.structure

import com.atlassian.query.Query
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.row.StructureRow
import com.almworks.jira.structure.api.row.RowManager
import com.almworks.jira.structure.api.item.ItemIdentity
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.util.JiraComponents
import com.almworks.integers.LongArray
import com.almworks.integers.LongIterator
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.label.LabelManager
import com.atlassian.jira.issue.worklog.WorklogManager
import com.atlassian.jira.issue.IssueManager

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version =
'16.10.0')

@WithPlugin("com.almworks.jira.structure")

@PluginModule
StructureComponents structureComponents

def plugin =
com.atlassian.jira.component.ComponentAccessor.pluginAccessor.getPlugin(
'com.almworks.jira.structure')

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def permission = PermissionLevel.valueOf("ADMIN")
def helper = plugin.getModuleDescriptor('helper').module

def struct = structureManager.getStructuresByName("test", permission)[0]
def structureName = struct.getName()
def forestSpec = ForestSpec.structure(struct.getId())
def forestSrc = forestService.getForestSource(forestSpec)
RowManager rowManager = structureComponents.getRowManager()
def forest = forestSrc.getLatest().getForest()

// this variable will store all the elements in our structure that are issues
(vs folders or generators).
```

```
LongArray onlyIssues = new LongArray()

// here we are iterating over all the rows of our structure to get the issues.
for (LongIterator ri : forest.getRows()) {
    StructureRow row = rowManager.getRow(ri.value())
    ItemIdentity itemId = row.getItemId()
    if (CoreIdentities.isIssue(itemId)){
        onlyIssues.add(itemId.getLongId())
    }
}

def wm = ComponentAccessor.getWorklogManager()
def issueManager = ComponentAccessor.getComponent(IssueManager)
def user = ComponentAccessor.jiraAuthenticationContext.getLoggedInUser()
def userToIssue = [:]
def seen = []

onlyIssues.each{ is ->
    def iss = issueManager.getIssueObject(is.value())
    def wl = wm.getByIssue(iss)

    def it = wl.each{
        def issue = iss.getKey()
        def author = it.getAuthorKey()
        def time = it.getTimeSpent()
        if (!seen.contains(author)){
            // put author in seen
 seen.push(author)
            def tempHM = [:]
            tempHM[issue] = time
            userToIssue[author] = tempHM
        } else {
            if(!userToIssue[author].containsKey(issue)){
                // create a new issue to time hashmap to add
 def tempHM = [:]
                tempHM[issue] = time
                userToIssue[author] += tempHM
            }
            else{
                userToIssue[author][issue] = userToIssue[author][issue] + time
            }
        }
    }
}

'<table><tr><th>Structure</th><th>User </th><th>Issue Worklogged (hours)</th></
tr>' +
        userToIssue.collect {'<tr><td>' + it.key + '</td><td><table>' +
                it.value.collect {'<tr><td>' + it.key + '</td> <td>' +
(it.value*0.000277778).toBigInteger() + '</td></tr>' }.join('') +
                '</table></td></tr>'}.join('') +
        '</table>'
```

3.14.3.11 Updating a field (ex. label) when checking all issues against a JQL query

This script will set the labels fields to a certain value for all the issues that satisfy a defined JQL query.

```
import com.atlassian.jira.bc.issue.search.SearchService
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.web.bean.PagerFilter
import com.atlassian.jira.issue.label.LabelManager

// list of elements to consider for matching

def searchlist = ["foo", "bar", "bar"]

def labelName = 'Partner'

def labelManager = ComponentAccessor.getComponent(LabelManager)
def user = ComponentAccessor.jiraAuthenticationContext.getLoggedInUser()
def searchService = ComponentAccessor.getComponent(SearchService)

for (searchItem in searchlist){
    def jqlSearch = "company = $searchItem"
    def parseResult = searchService.parseQuery(user, jqlSearch)
    if (parseResult.isValid()) {
        def searchResult = searchService.search(user, parseResult.getQuery(),
PagerFilter.getUnlimitedFilter())
        searchResult.issues.each{issue -> labelManager.addLabel(user, issue.id,
labelName, false)
        }
    }
}
```

3.14.4 Power Scripts for Jira

Power Scripts™ for Jira[104] provides custom Jira scripting and automation capabilities to automate workflows and complex tasks.

3.14.4.1 Key benefits to using Power Scripts with Structure

- Add scripted content as a column within the structure
- Use scripts to display specific issue dependencies within the structure

Learn More: Power Scripts™ + Structure for Jira Combined Simplify Dependency Management[105]

104 https://marketplace.atlassian.com/apps/43318/power-scripts-jira-script-automation?hosting=server&tab=overview
105 https://www.cprimeapps.com/power-scripts-structure-for-jira-combined-simplify-dependency-management/

3.14.5 Timesheets by Tempo

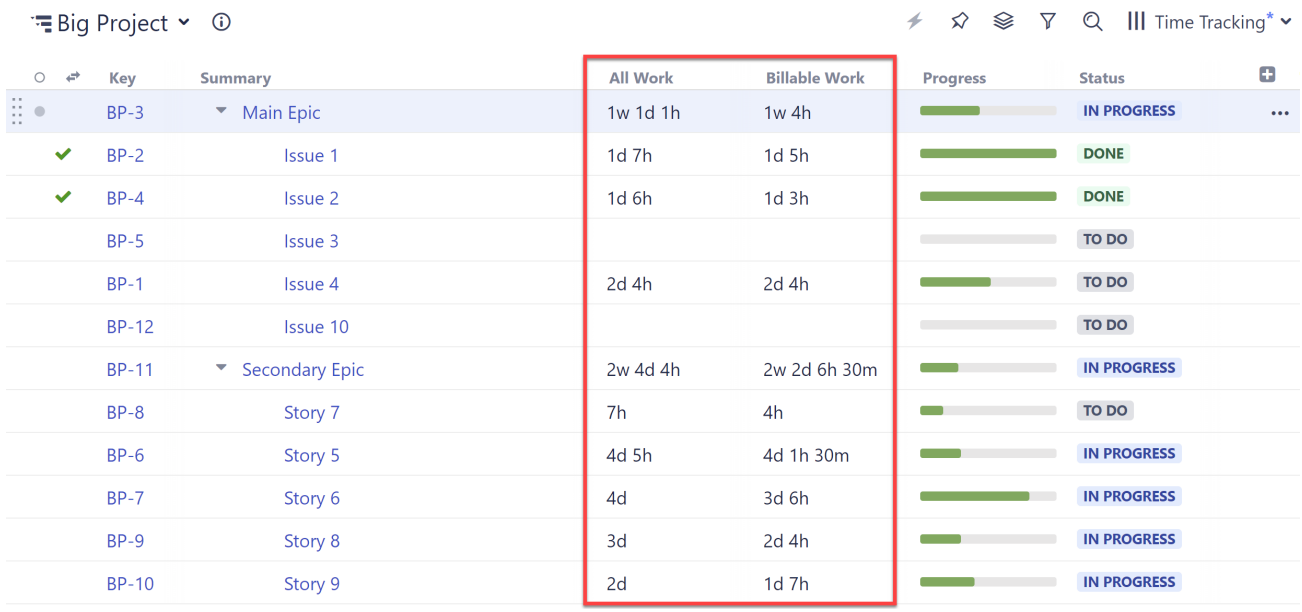
If you're using the Timesheets by Tempo app[106],

our integration allows you to:

- Add Tempo Work Logged data as columns in your structure
- Use worked logged data in formulas, generators, and transformations
- Organize your structure data based on Tempo accounts and Tempo teams

3.14.5.1 Adding a Tempo Work Logged Column

Adding a custom Tempo Work Logged column to a structure allows you to easily track work time and billable hours alongside your project data, and aggregate those values up your hierarchy.



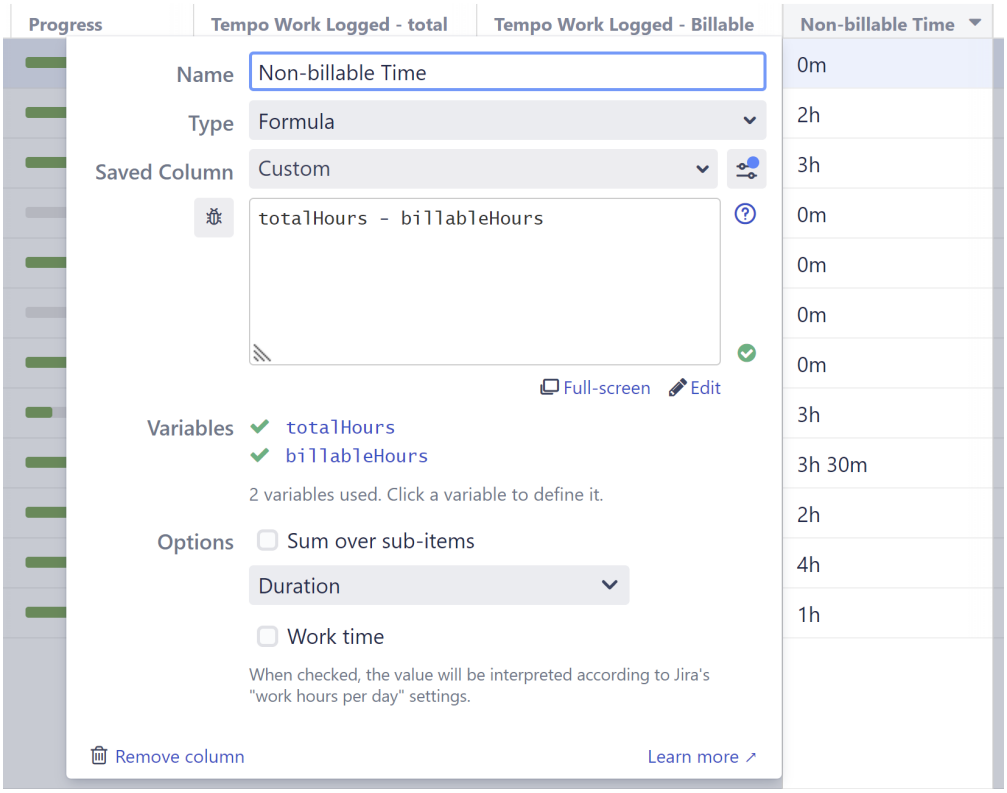
Key	Summary	All Work	Billable Work	Progress	Status
BP-3	Main Epic	1w 1d 1h	1w 4h		IN PROGRESS
BP-2	Issue 1	1d 7h	1d 5h		DONE
BP-4	Issue 2	1d 6h	1d 3h		DONE
BP-5	Issue 3				TO DO
BP-1	Issue 4	2d 4h	2d 4h		TO DO
BP-12	Issue 10				TO DO
BP-11	Secondary Epic	2w 4d 4h	2w 2d 6h 30m		IN PROGRESS
BP-8	Story 7	7h	4h		TO DO
BP-6	Story 5	4d 5h	4d 1h 30m		IN PROGRESS
BP-7	Story 6	4d	3d 6h		IN PROGRESS
BP-9	Story 8	3d	2d 4h		IN PROGRESS
BP-10	Story 9	2d	1d 7h		IN PROGRESS

Learn more: Tempo Work Logged Column(see page 491)

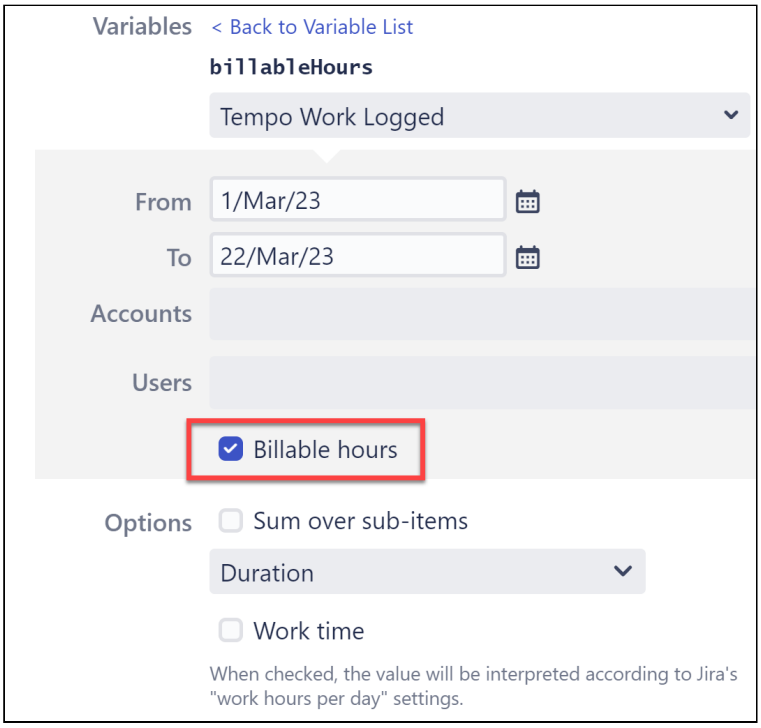
3.14.5.2 Using Work Logged Data in Formulas

106 https://marketplace.atlassian.com/apps/6572/tempo-timesheets-time-tracking-report

You can use work logged data within a formula to do calculations or comparisons based on your work logged values.



When mapping a variable to Tempo Work Logged, you can specify the same values as in the Tempo Work Logged Column(see page 491). In the example above, we used identical settings for both variables, except that for "billableHours" we checked the "Billable hours" option.



3.14.5.3 Grouping by Tempo Accounts

1. Add tasks, either manually or using Generators(see page 140).
2. Group tasks using the **Tempo Account Grouper**. In the Automation menu, select **Automation | Group | Tempo Account...**
3. Add Structure columns to view the necessary data, such as Progress or Time Spent.

In the Time Spent column options, make sure the **Sum over sub-items** option is checked. This will aggregate your time data for each account.

> The "Tempo Account" option in Structure is for the Account Custom Field. The Account Work Attribute is not available at this time.

Archived/Closed Accounts

When you add the Tempo Account grouper, you can select whether or not to display closed or archived accounts. To change this setting afterward, simply double-click the grouper in your structure.

3.14.5.4 Grouping by Teams

To group issues by Tempo Teams, use the Group by Attribute(see page 175) generator.

3.14.6 Xporter

Xporter[107] allows you easily export Jira issues to PDF, Excel and Word.

3.14.6.1 Key benefits to using Xporter with Structure

- Export structures directly from the Structure board
- Export structures from the issue screen
- Export children on any row
- Extract data from specific rows
- Export the children count for issues in each row

Learn more: Xporter - Structure Integration[108]

3.14.7 Xray Test Management for Jira

Xray[109] provides automated test management for Jira.

3.14.7.1 Key benefits to using Xray with Structure

- Include Xray Test and Test Sets within the Structure hierarchy
- Add/remove Tests and Test Sets by dragging them within the structure
- Visualize and manage associations between Tests and Requirements within Structure

Learn more: Xray - Structure Integration[110]

107 https://marketplace.atlassian.com/apps/891368/xporter-export-issues-from-jira
108 https://confluence.xpand-it.com/display/public/XPORTER/Structure+for+Jira+-+Projects+at+Scale
109 https://marketplace.atlassian.com/apps/1211769/xray-test-management-for-jira?hosting=server&tab=overview
110 https://docs.getxray.app/display/XRAY/Integration+with+Structure

4 Build Structures for Your Role

Build custom structures to accomplish your daily tasks.

4.1 Product Owner

Cost Calculations for Issues and Projects

Design a structure for calculating what it will cost to complete each Story, Epic, or other grouping of issues, based on the amount of time remaining and the hourly rate of the assignee.

4.1.1 Requirements

For this method to be successful, you need:

- All stories and tasks assigned to specific Jira users
- A custom User property[111] containing each user's hourly rate of pay (we've named ours "cost")

4.1.2 Step 1: Build a Hierarchy

Build a structure with an Epic > Story > Sub-task hierarchy:

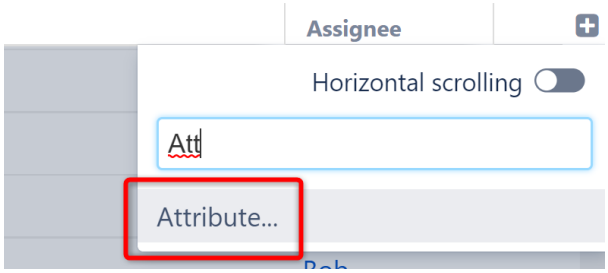
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add epics: **Automation | Insert | JQL Query** and enter the following JQL query: `issuetype = Epic`
    a. To limit the epics to specific projects or other variables, add additional specifications (Example: `AND project = "My Epics"`)
3. Add stories: **Automation | Extend | Stories under Epics...**
4. Add sub-tasks (optional): **Automation | Extend | Sub-tasks...**

There are many ways to build your hierarchy, depending on what values you want to track. For more ideas, check out some of our other Task Articles(see page 637) or our Generators(see page 140) docs.

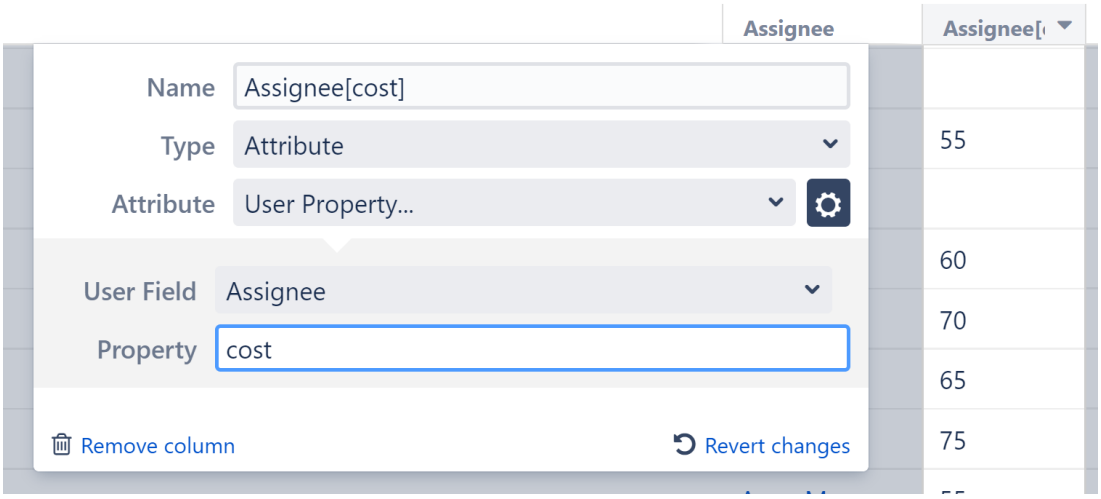
111 https://confluence.atlassian.com/adminjiraserver/create-edit-or-remove-a-user-938847025.html

4.1.3 Step 2: Add a Column for the Custom User Property

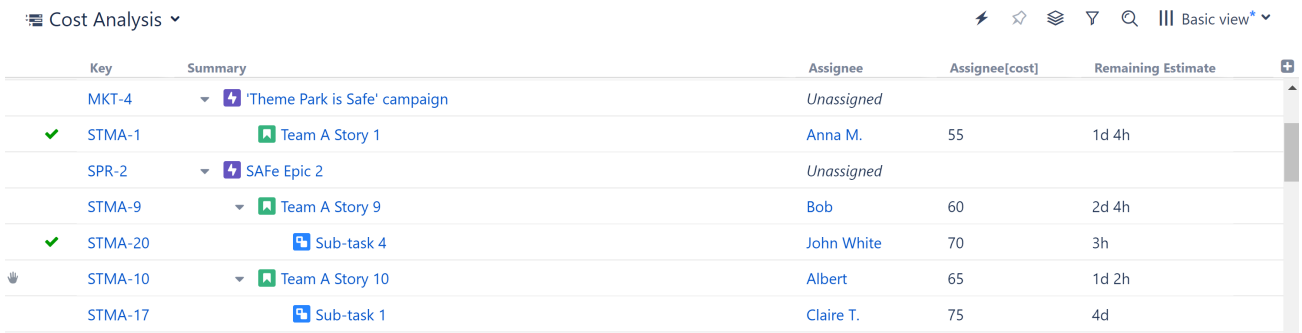
1. Open the Add Column menu (**+**) and select **Attribute...**



2. Under the column properties, enter the following values:
   - **Attribute:** User Property...
   - **User Field:** Assignee
   - **Property:** Enter the key for the user property where you store each user's hourly pay rate - ours is called "cost"



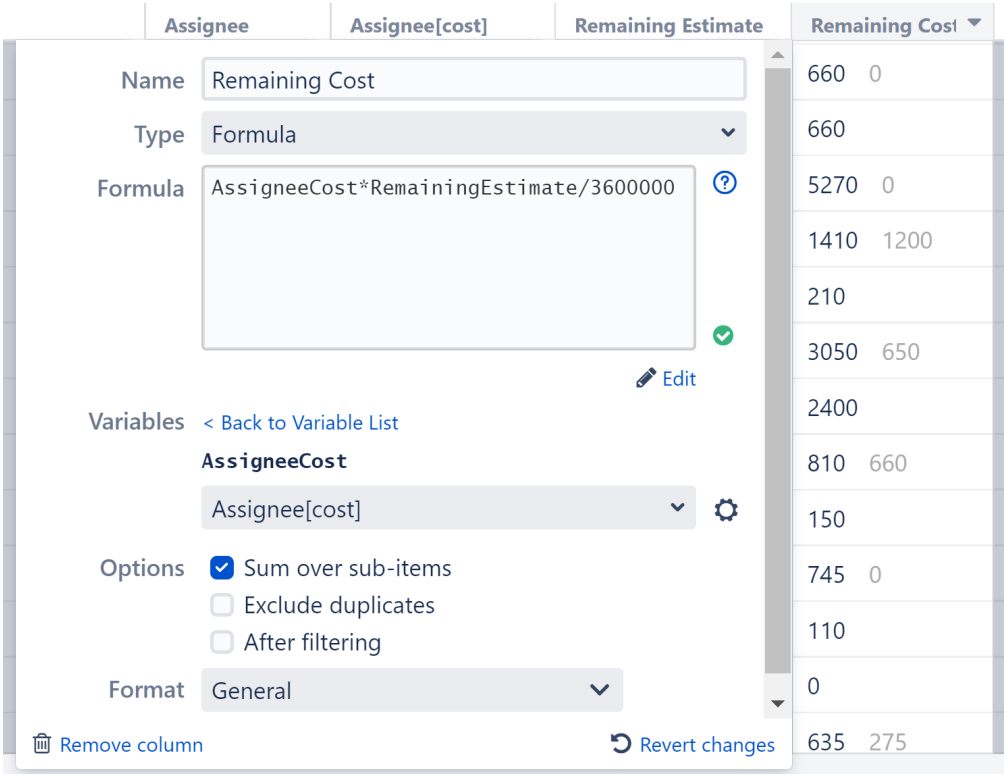
You should now have an **Assignee(cost)** column that displays the values from our custom user property:



4.1.4 Step 3: Calculate the Cost to Finish Each Issue (epic/story/etc.)

Add a formula column(see page 604) to calculate the cost to finish each issue in our structure. Enter the following:

- Formula: `AssigneeCost*RemainingEstimate/3600000`
- Variables: map AssigneeCost to the custom "cost" property we discussed in Step 2 - see Columns as Variables(see page 233) for more details
- Check **Sum over sub-items** so the cost is aggregated up the hierarchy
- If issues or tasks appear multiple times within an epic (or other grouping), check **Exclude deplicates** to avoid an inflated total costs



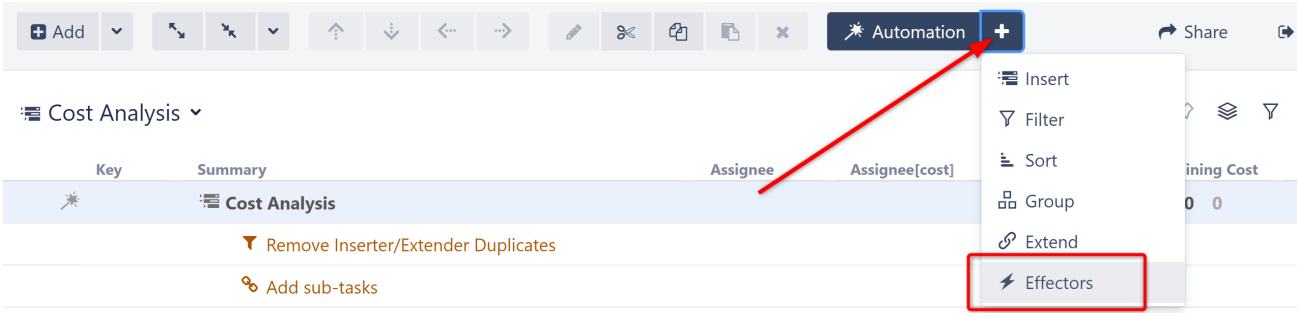
4.1.5 Bonus - Make This Information Visible Outside of Structure

Using Effectors(see page 179), we can write the values from our formula to a custom Jira field, making them visible to users outside of Structure.

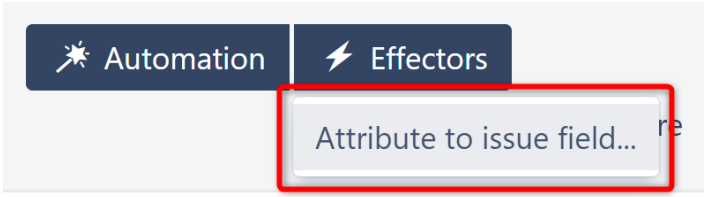
Before we start, you'll need to make sure you have a custom field where the Cost-to-Complete values can be stored. We created a custom field called "FinishCost".

4.1.5.1 Add an Effector

Open the Automation menu and select **Effectors**.

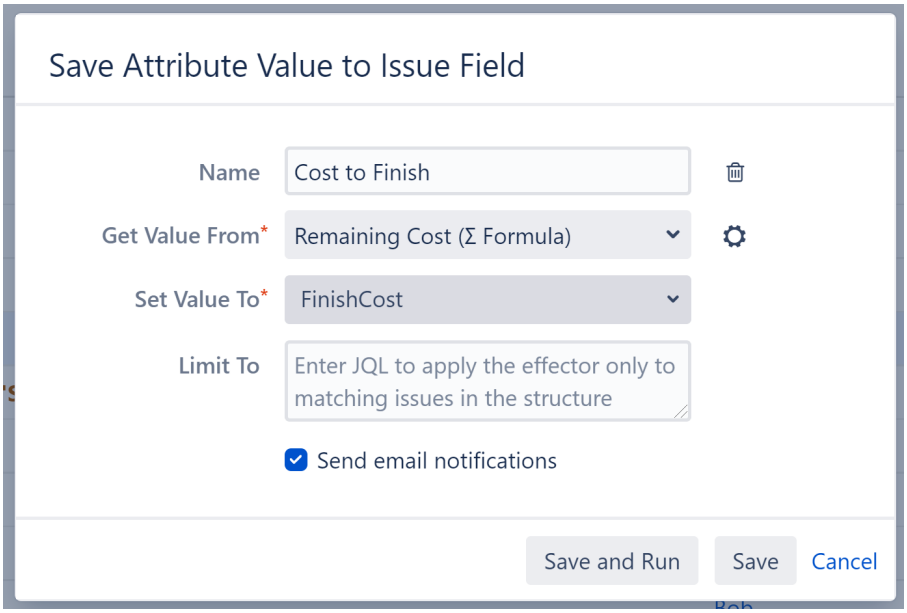


Choose **Attribute to issue field...**



On the Effector settings screen:

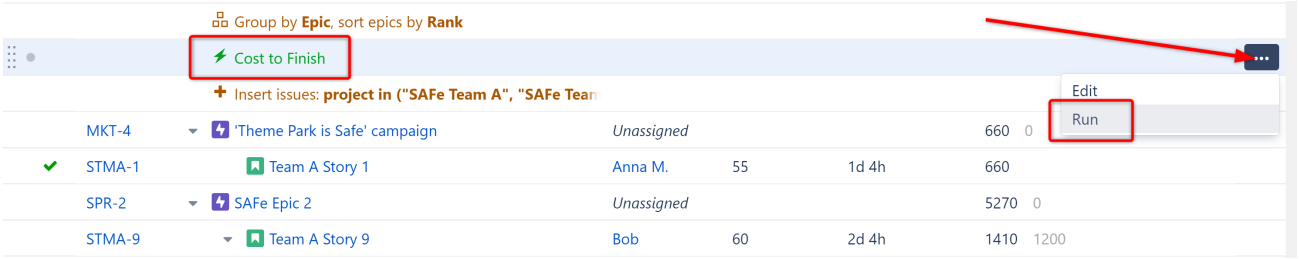
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Get Value From: select the formula column we added in Step 2
- Set Value To: select the custom Jira field you want to write the cost-to-complete values to
- Limit To: If you only want to write cost data for certain issues or issue types (for example, just for epics), enter the appropriate JQL. We've left this section blank, because we want to write values for all issues.
- Select whether email notifications should be sent when the Effector writes values to Jira



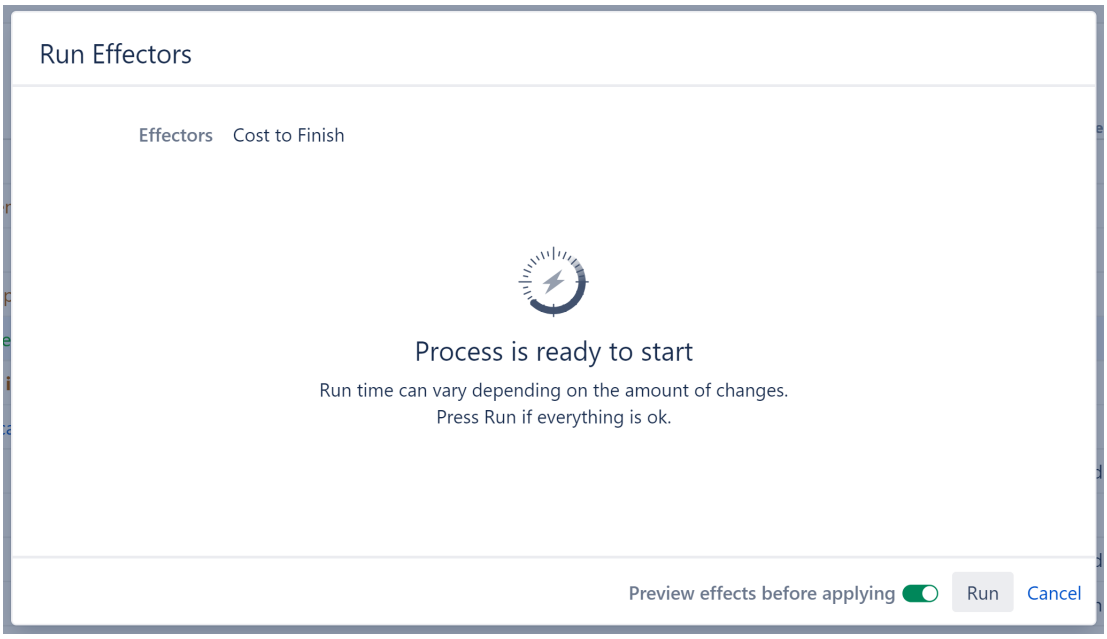
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

4.1.5.2 Run the Effector

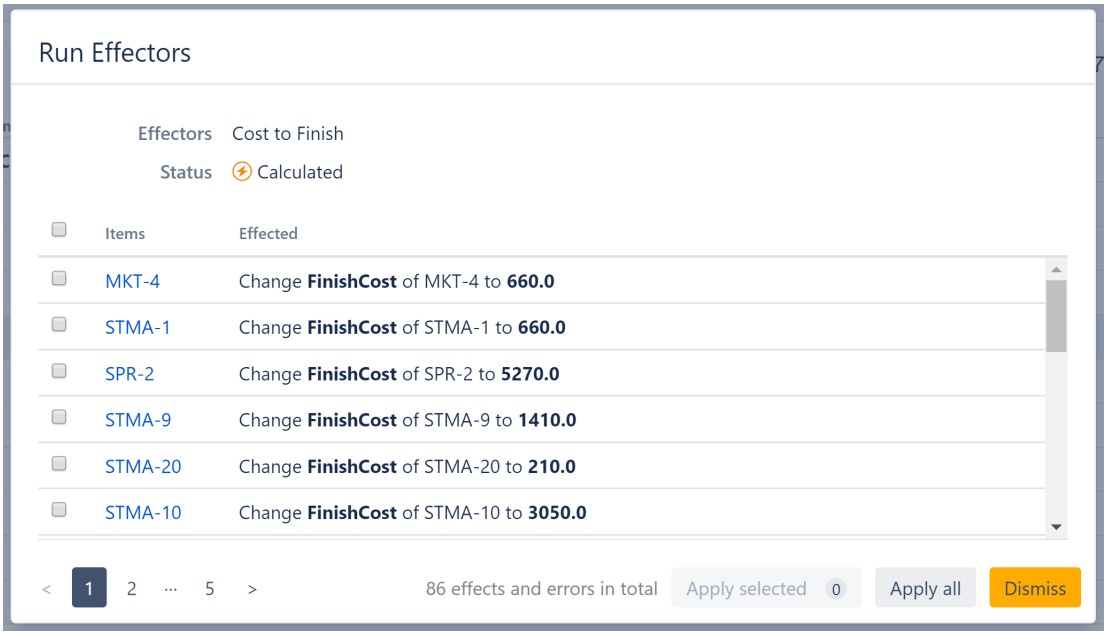
You can run an Effector directly from its settings screen (see above), or you can locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.



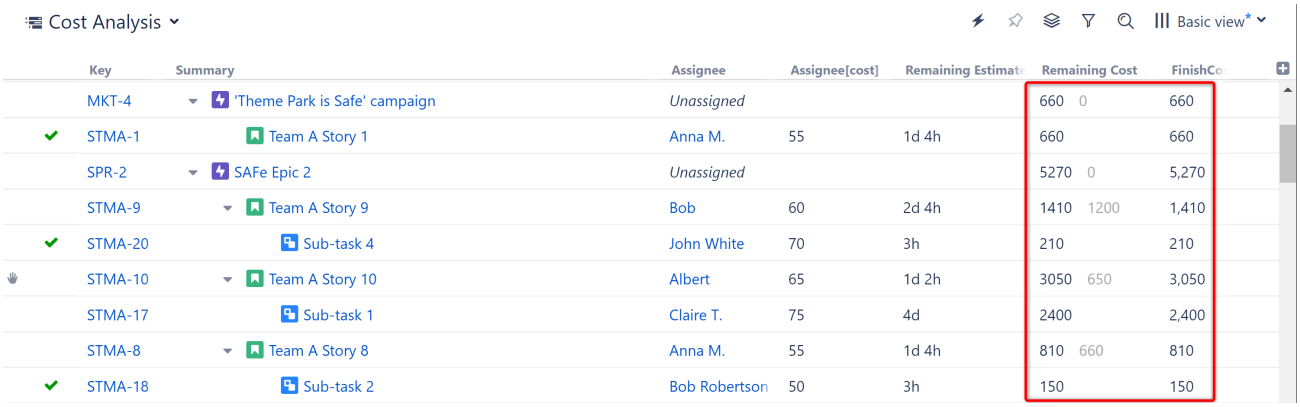
You have the option to **Preview effects before applying**. This allows you to view and approve every change that will be made by the Effector. Effectors update live Jira data, so we highly recommend using the preview option. Click **Run** to begin.



Once the preview is finished, you will see a list of changes that will be made by the Effector. You can select which changes you want made, or click **Apply All** to apply all changes.



Once you apply the selected changes, the cost values from your formula column will be written to the custom Jira field, where they can be viewed outside of Structure.



Custom Reports

With Structure, you can easily create a visual overview of your projects – and visualize all the data you need to track on a single screen.

4.1.5.3 Step 1: Build Your Structure

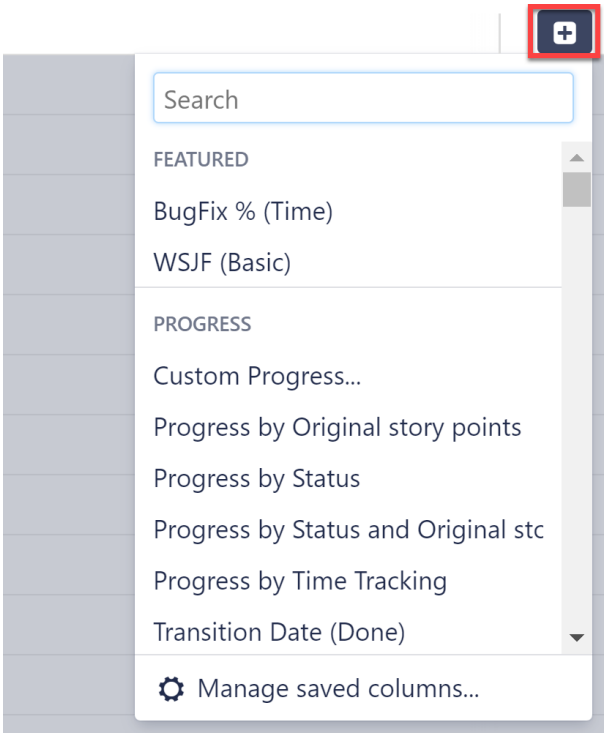
To create a new structure:

- Go to the Jira menu and select **Structure | Create Structure**
- Add the issues you want to track – you can add them manually or automatically using **Automation | Insert**
- Add related issues - you can do this using **Automation | Group** and **Automation | Extend**

4.1.5.4 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate
- Progress[112] - track status based on issue progress, percent completion, or resolution
- Tempo Work Logged[113] - track how long your team is spending on each issue
- Formulas[114] - create your own formulas to compare fields or create a visual report
- Time in Status[115] - see how much time issues spend in a particular status
- Last Comment[116] - view the latest comment for each issue



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

Release Management

With Structure, you can track all the issues for upcoming releases – even if they're from different projects – and visualize all the data you need to track on a single screen.

4.1.5.5 Step 1: Build a Release Management Structure

To create a structure for release management:

1. Go to the Jira menu and select **Structure | Create Structure**
2. Add the issues you want to track – you can add them manually or automatically using **Automation | Insert**

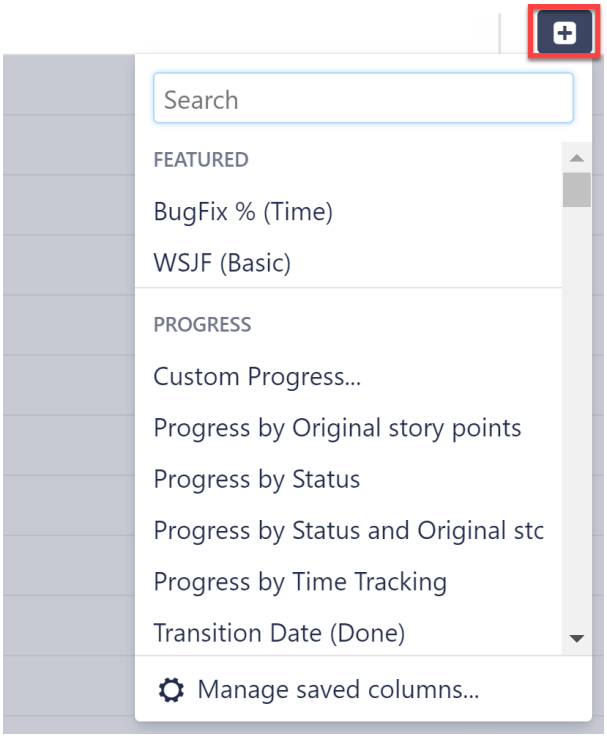
112 https://wiki.almworks.com/display/structure/Progress+Column
113 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
114 https://almworks.com/structure/help/column/formula
115 https://wiki.almworks.com/display/structure/Time+in+Status+Column
116 https://almworks.com/structure/help/column/last-comment

3. Group issues by fix version: **Automation | Group | type "Fix Version/s"** *(Tip: If you use the same names in different projects, use **"Version name..."** instead.)*
4. Open the Summary settings and select **Show Sprint and Version attributes**

4.1.5.6 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate
- Progress[117] - track status based on issue progress, percent completion, or resolution
- Tempo Work Logged[118] - track how long your team is spending on each issue
- Formulas[119] - create your own formulas to compare fields or create a visual report
- Time in Status[120] - see how much time issues spend in a particular status
- Last Comment[121] - view the latest comment for each issue



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

Resource Allocation

When team members are working on multiple projects, it can be difficult to track how much work everyone is doing. Using Structure, you can track allocation across projects and reassign work on the fly.

---

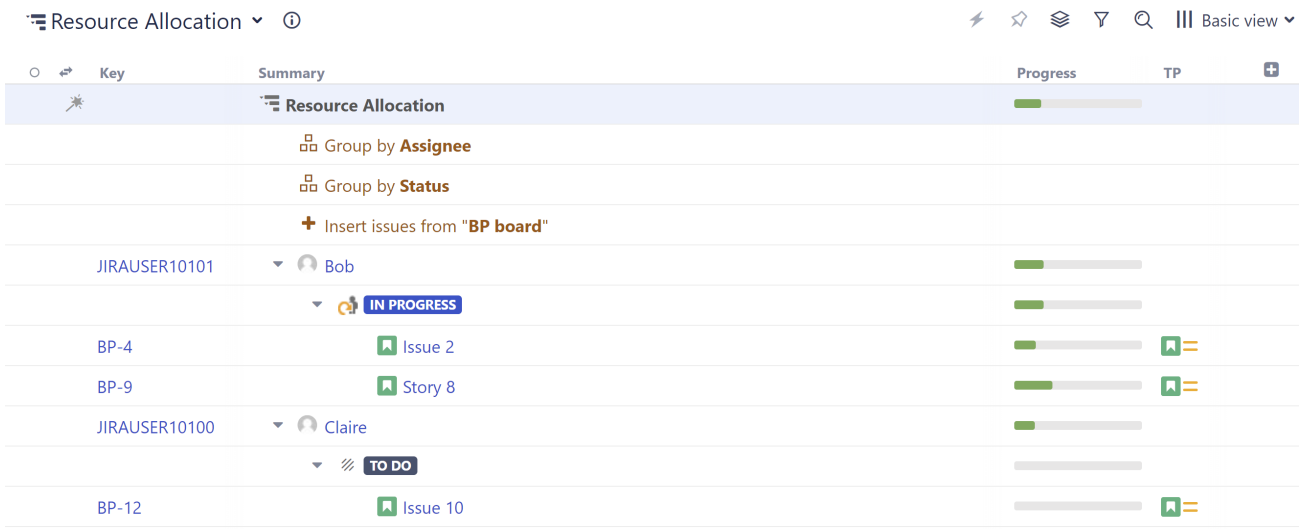
117 https://wiki.almworks.com/display/structure/Progress+Column
118 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
119 https://almworks.com/structure/help/column/formula
120 https://wiki.almworks.com/display/structure/Time+in+Status+Column
121 https://almworks.com/structure/help/column/last-comment

4.1.5.7 Step 1: Build a Resource Allocation Structure

Insert issues from every project/board the team members work on, and then group them by Progress and Assignee.

To create a new structure:

1. Go to the Jira menu and select **Structure | Create Structure**
2. Add all the issues: **Automation | Insert**
    - Select **Agile Board** to add issues from select boards
    - Select **JQL Query** to add issues from projects, using JQL
3. Group issues by Status: **Automation | Group | type "Status"**
4. Group issues by Assignee: **Automation | Group | type "Assignee"**



If the status is appearing above the assignee, drag the "Group by Status" row under the "Group by Assignee" row.

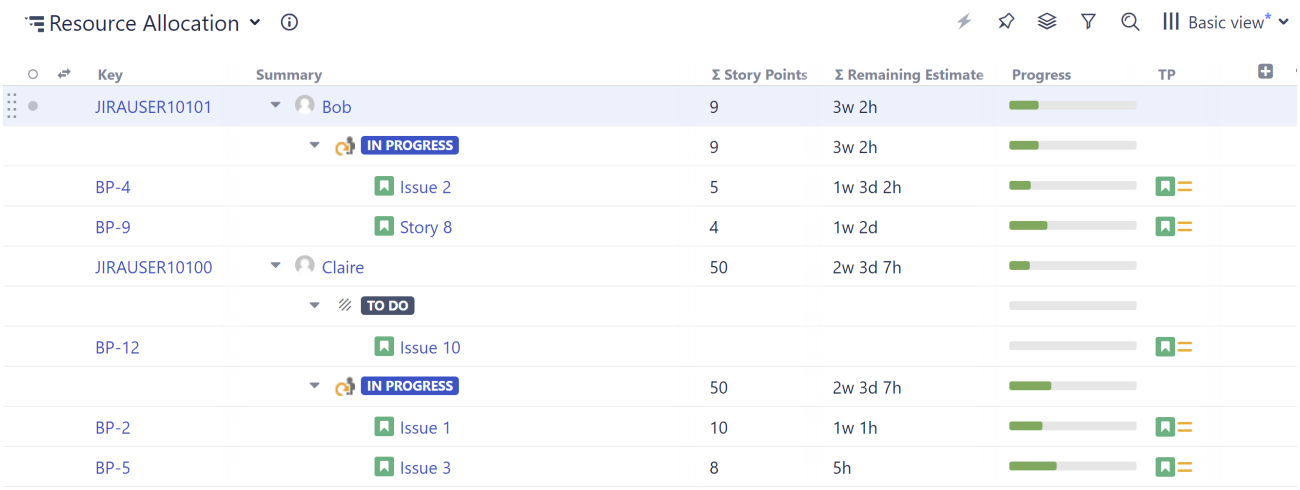
This will create a grouping for every team member with at least one issue in the current structure assigned to them. If someone doesn't have any issues assigned to them, they won't appear here. To fix this, simply assign an unassigned issue to that team member, and a new grouping will appear.

4.1.5.8 Step 2: Add Columns to Compare Workloads

Structure columns allow you to visualize multiple data points in a single view. We recommend one or more of these:

- Status
- ∑ Story Points (if you're using them)
- ∑ Original Estimate
- ∑ Remaining Estimate
- A custom formula(see page 463) to calculate the remaining effort

Using a Totals columns, the values for individual issues are aggregated up to their assignee, so you can evaluate workloads at a glance.



4.1.5.9 Step 3: Drag Issues to Reassign

To assign/reassign tasks, simply drag issues from one assignee grouping to another.

4.1.5.10 Optional Enhancements:

- To allocate resources at the group level, group issues by a custom Team field instead of Assignee.
- Add a WSJF column(see page 257) to prioritize issues, and select Sum over sub-items to balance priorities across resources.
- Manage allocation across sprints by adding a Group by Sprint generator. (Haven't assigned sprints yet? Check out Sprint Planning with Structure(see page 657)!)
- Consider using Structure.Gantt[122] - it can identify and resolve overallocation automatically!

Sprint Planning

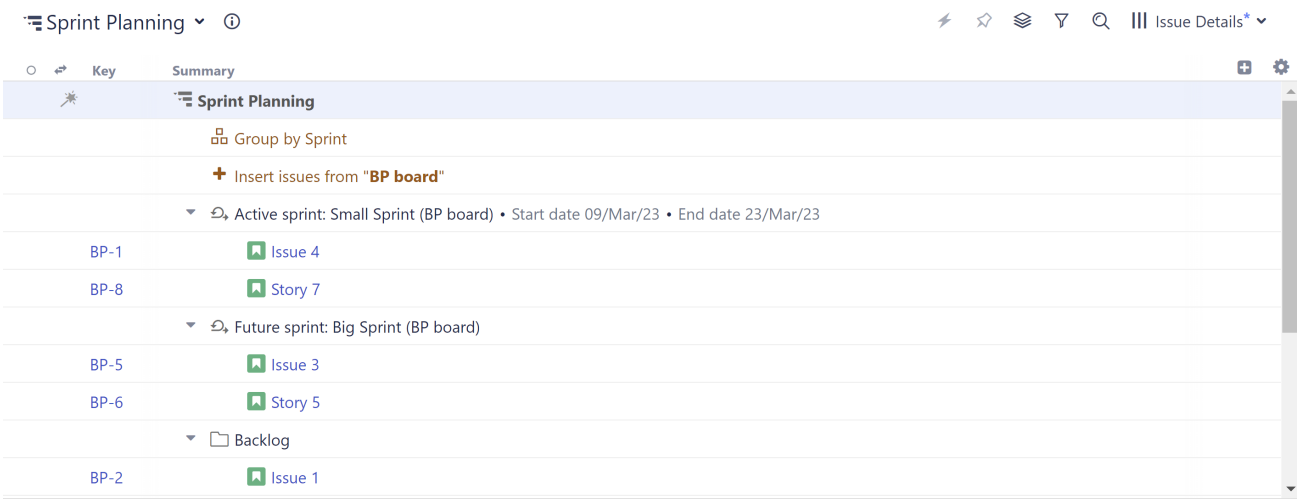
With Structure, you can track all the issues for upcoming sprints – even if they're from different projects – and visualize all the data you need to track on a single screen. You can even aggregate that data for each sprint.

4.1.5.11 Step 1: Build a Sprint Planning Structure

To create a new structure:

1. Go to the Jira menu and select **Structure | Create Structure**
2. Add all the issues from a Board: **Automation | Insert | Agile Board**
3. Group by sprints: **Automation | Group | type "Sprint..."**
4. Open the Summary settings and select **"Show Sprint and Version attributes"**:

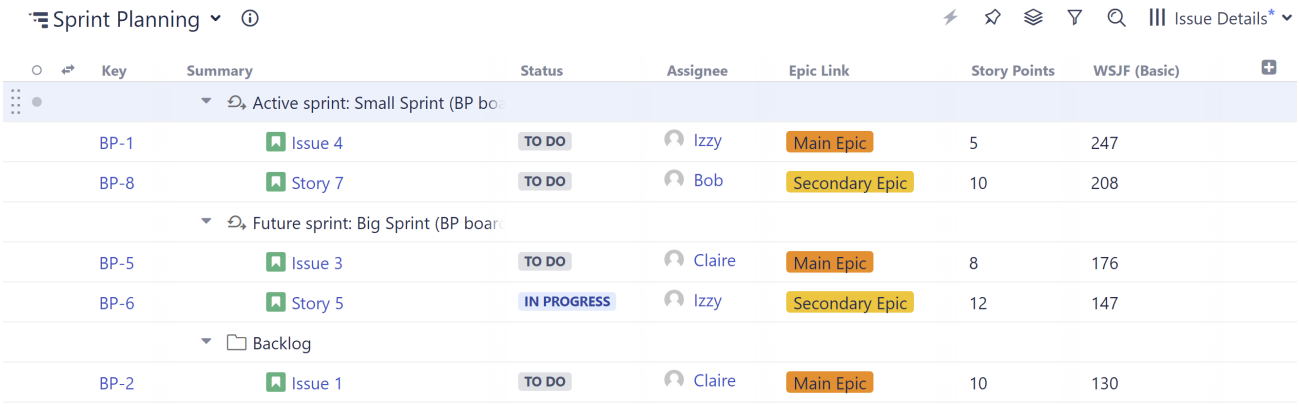
122 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale



4.1.5.12 Step 2: Add Data

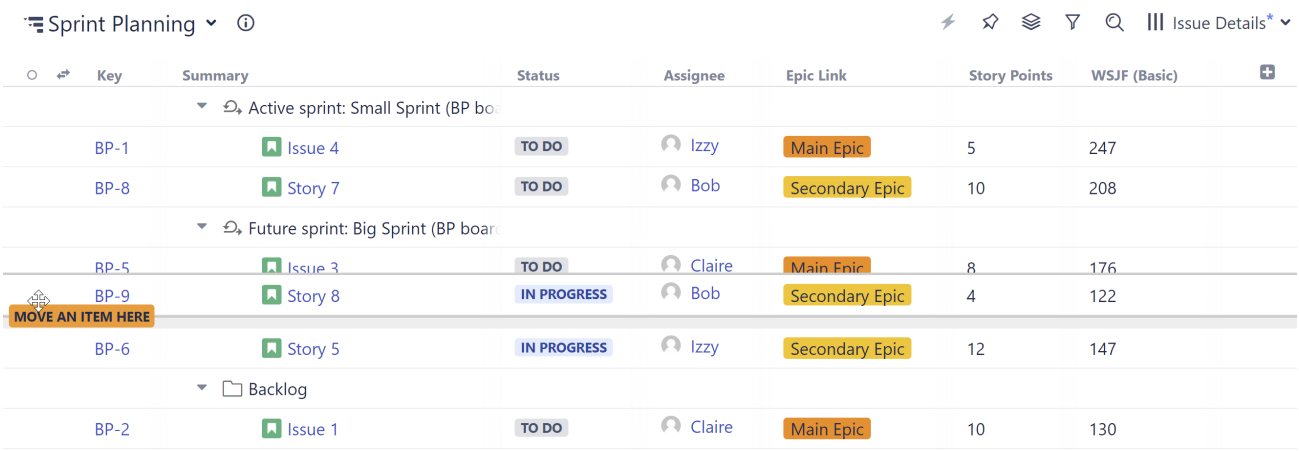
Next, add the data you'll use to compare issue priority:

1. Click the **+** button at the top-right corner of the structure
2. Select the fields or attributes you want to compare - we recommend:
    - Status
    - Assignee
    - Epic Link
    - Story Points (if you're using them)
    - WSJF (or any other metric your team uses to prioritize issues or assess business value)
3. Click the header of any column to sort your issues by that value.



4.1.5.13 Step 3: Assign Issues

To assign issues to a sprint, drag it from the backlog to the appropriate sprint. You can also reassign issues by dragging them from one sprint to another.



4.1.5.14 Optional Enhancements

- Aggregate values for each sprint: when adding a column (such as Story Points), select **Sum over sub-items**
- Group by assignee under each sprint to see how the work is divided across your team: **Automation | Group | Assignee**
- Hide closed sprints: **Automation | Filter | Hide Closed Sprints**
- Use Structure.Gantt[123] to view any blocking dependencies between issues (to help prioritize them)

4.2 Program Manager

Cost Calculations for Issues and Projects

Design a structure for calculating what it will cost to complete each Story, Epic, or other grouping of issues, based on the amount of time remaining and the hourly rate of the assignee.

4.2.1 Requirements

For this method to be successful, you need:

- All stories and tasks assigned to specific Jira users
- A custom User property[124] containing each user's hourly rate of pay (we've named ours "cost")

4.2.2 Step 1: Build a Hierarchy

Build a structure with an Epic > Story > Sub-task hierarchy:

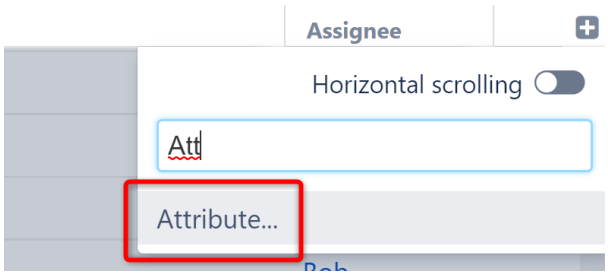
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add epics: **Automation | Insert | JQL Query** and enter the following JQL query: `issuetype = Epic`
    a. To limit the epics to specific projects or other variables, add additional specifications (Example: `AND project = "My Epics"`)
3. Add stories: **Automation | Extend | Stories under Epics...**
4. Add sub-tasks (optional): **Automation | Extend | Sub-tasks...**

123 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale
124 https://confluence.atlassian.com/adminjiraserver/create-edit-or-remove-a-user-938847025.html

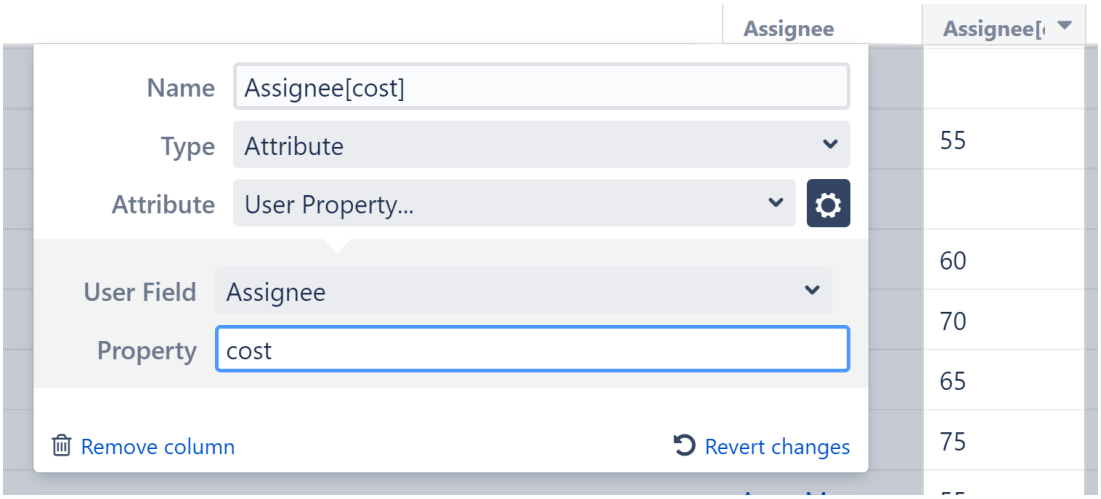
There are many ways to build your hierarchy, depending on what values you want to track. For more ideas, check out some of our other Task Articles(see page 637) or our Generators(see page 140) docs.

4.2.3 Step 2: Add a Column for the Custom User Property

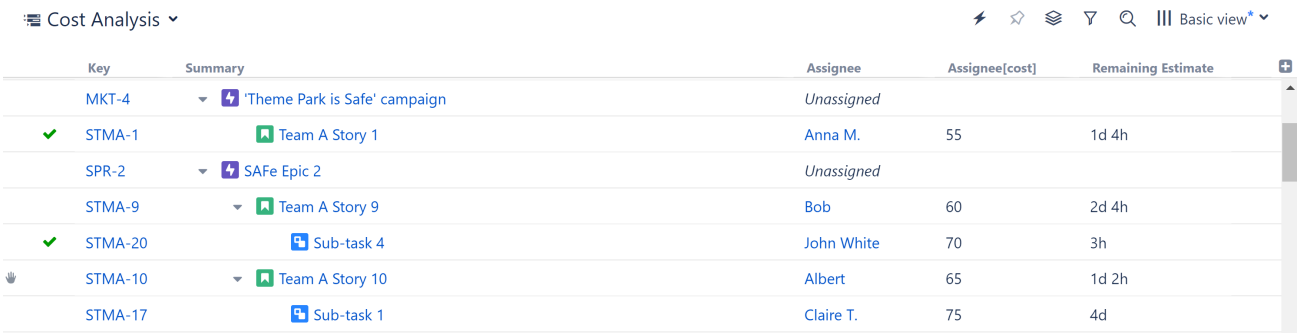
1. Open the Add Column menu (**+**) and select **Attribute...**



2. Under the column properties, enter the following values:
   - **Attribute:** User Property...
   - **User Field:** Assignee
   - **Property:** Enter the key for the user property where you store each user's hourly pay rate - ours is called "cost"



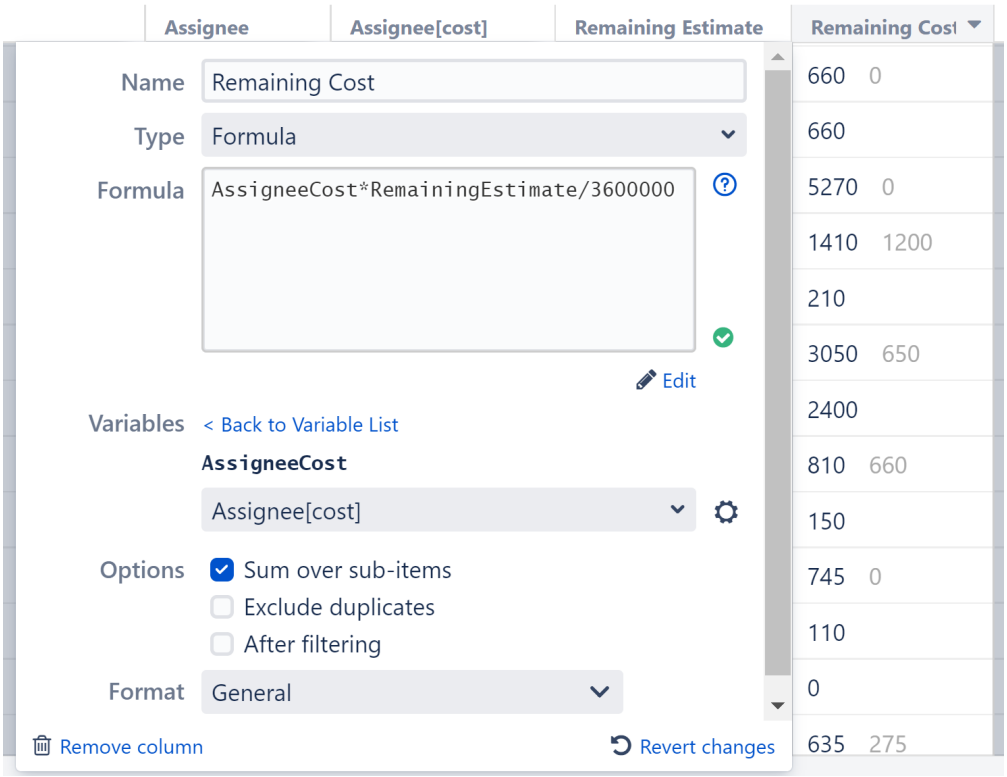
You should now have an **Assignee(cost)** column that displays the values from our custom user property:



4.2.4 Step 3: Calculate the Cost to Finish Each Issue (epic/story/etc.)

Add a formula column(see page 615) to calculate the cost to finish each issue in our structure. Enter the following:

- Formula: `AssigneeCost*RemainingEstimate/3600000`
- Variables: map AssigneeCost to the custom "cost" property we discussed in Step 2 - see Columns as Variables(see page 233) for more details
- Check **Sum over sub-items** so the cost is aggregated up the hierarchy
- If issues or tasks appear multiple times within an epic (or other grouping), check **Exclude deplicates** to avoid an inflated total costs



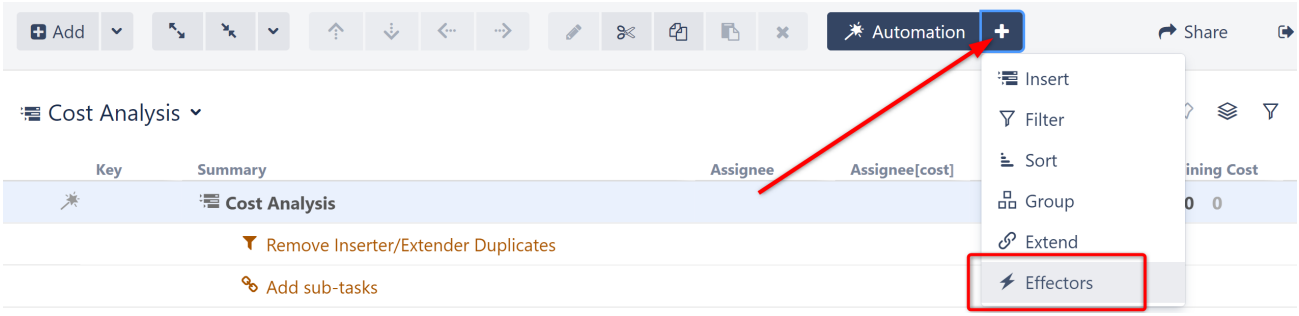
4.2.5 Bonus - Make This Information Visible Outside of Structure

Using Effectors(see page 179), we can write the values from our formula to a custom Jira field, making them visible to users outside of Structure.

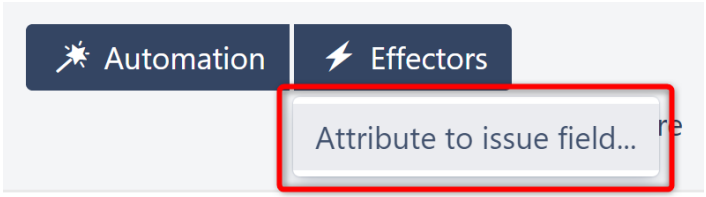
Before we start, you'll need to make sure you have a custom field where the Cost-to-Complete values can be stored. We created a custom field called "FinishCost".

4.2.5.1 Add an Effector

Open the Automation menu and select **Effectors**.

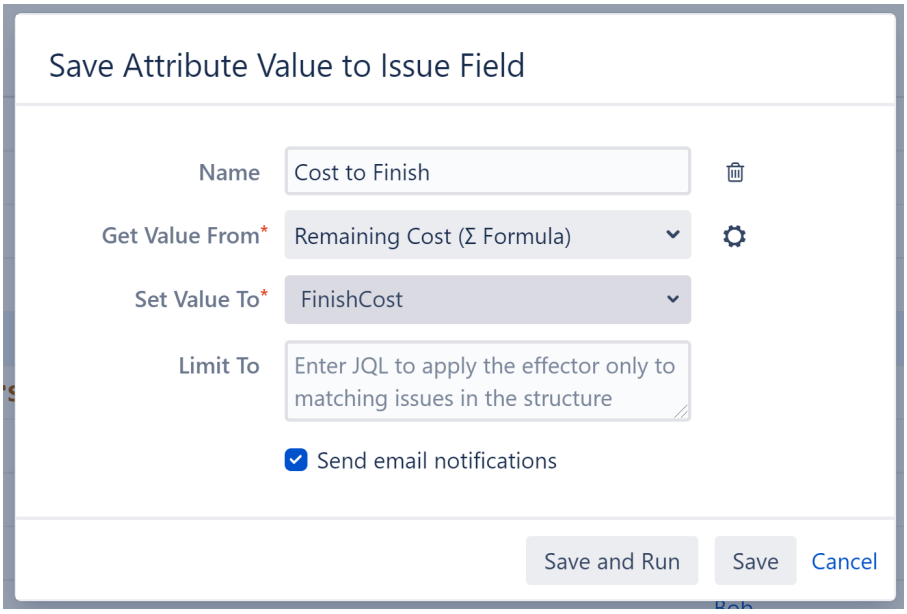


Choose **Attribute to issue field...**



On the Effector settings screen:

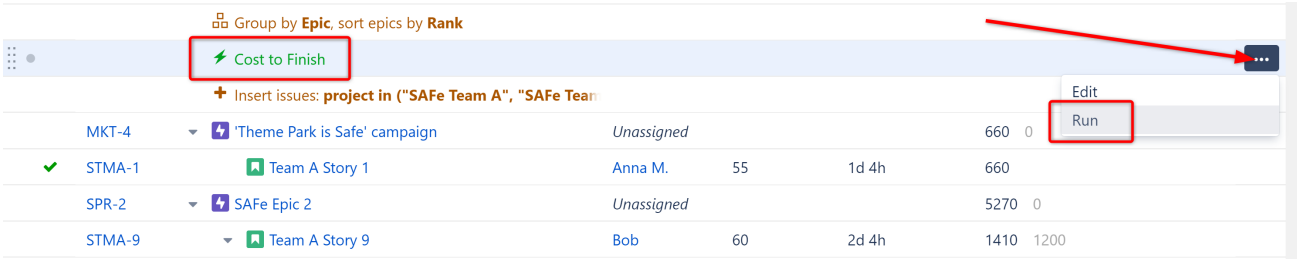
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Get Value From: select the formula column we added in Step 2
- Set Value To: select the custom Jira field you want to write the cost-to-complete values to
- Limit To: If you only want to write cost data for certain issues or issue types (for example, just for epics), enter the appropriate JQL. We've left this section blank, because we want to write values for all issues.
- Select whether email notifications should be sent when the Effector writes values to Jira



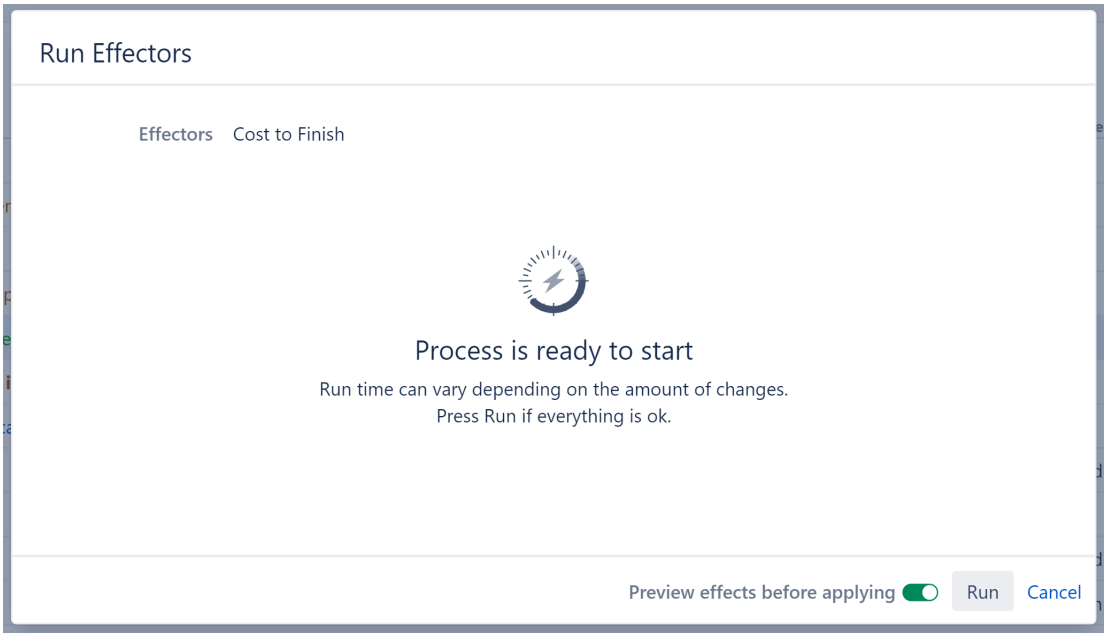
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

4.2.5.2 Run the Effector

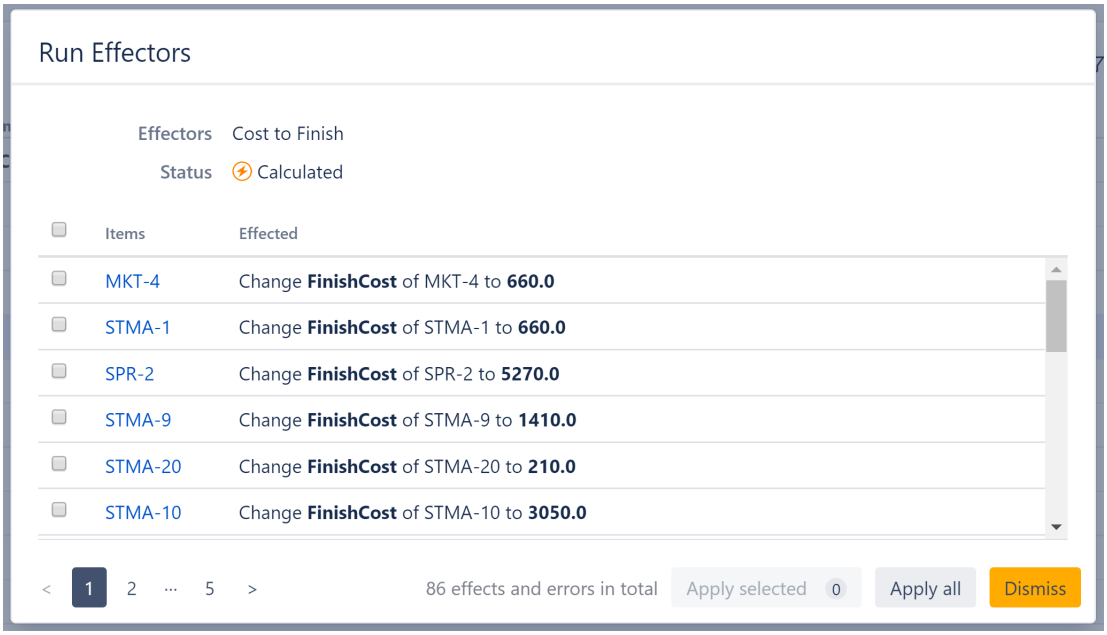
You can run an Effector directly from its settings screen (see above), or you can locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.



You have the option to **Preview effects before applying**. This allows you to view and approve every change that will be made by the Effector. Effectors update live Jira data, so we highly recommend using the preview option. Click **Run** to begin.



Once the preview is finished, you will see a list of changes that will be made by the Effector. You can select which changes you want made, or click **Apply All** to apply all changes.



Once you apply the selected changes, the cost values from your formula column will be written to the custom Jira field, where they can be viewed outside of Structure.

	Key	Summary	Assignee	Assignee[cost]	Remaining Estimate	Remaining Cost	FinishCo
	MKT-4	'Theme Park is Safe' campaign	*Unassigned*			660 0	660
✔	STMA-1	Team A Story 1	Anna M.	55	1d 4h	660	660
	SPR-2	SAFe Epic 2	*Unassigned*			5270 0	5,270
	STMA-9	Team A Story 9	Bob	60	2d 4h	1410 1200	1,410
✔	STMA-20	Sub-task 4	John White	70	3h	210	210
	STMA-10	Team A Story 10	Albert	65	1d 2h	3050 650	3,050
	STMA-17	Sub-task 1	Claire T.	75	4d	2400	2,400
	STMA-8	Team A Story 8	Anna M.	55	1d 4h	810 660	810
✔	STMA-18	Sub-task 2	Bob Robertson	50	3h	150	150

Custom Reports

With Structure, you can easily create a visual overview of your projects – and visualize all the data you need to track on a single screen.

4.2.5.3 Step 1: Build Your Structure

To create a new structure:

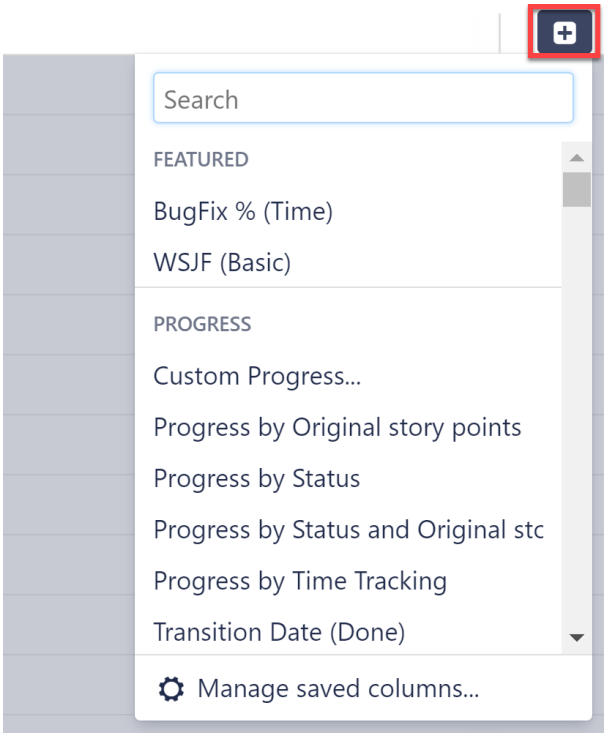
- Go to the Jira menu and select **Structure | Create Structure**
- Add the issues you want to track – you can add them manually or automatically using **Automation | Insert**
- Add related issues - you can do this using **Automation | Group** and **Automation | Extend**

4.2.5.4 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate
- Progress[125] - track status based on issue progress, percent completion, or resolution
- Tempo Work Logged[126] - track how long your team is spending on each issue
- Formulas[127] - create your own formulas to compare fields or create a visual report
- Time in Status[128] - see how much time issues spend in a particular status
- Last Comment[129] - view the latest comment for each issue

125 https://wiki.almworks.com/display/structure/Progress+Column
126 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
127 https://almworks.com/structure/help/column/formula
128 https://wiki.almworks.com/display/structure/Time+in+Status+Column
129 https://almworks.com/structure/help/column/last-comment



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

SAFe Planning

Everyone uses the Scaled Agile Framework (SAFe) a bit differently, but Structure is highly customizable and easy to tailor to your specific SAFe needs.

4.2.5.5 Step 1: Build a SAFe Structure

To create a structure for SAFe:

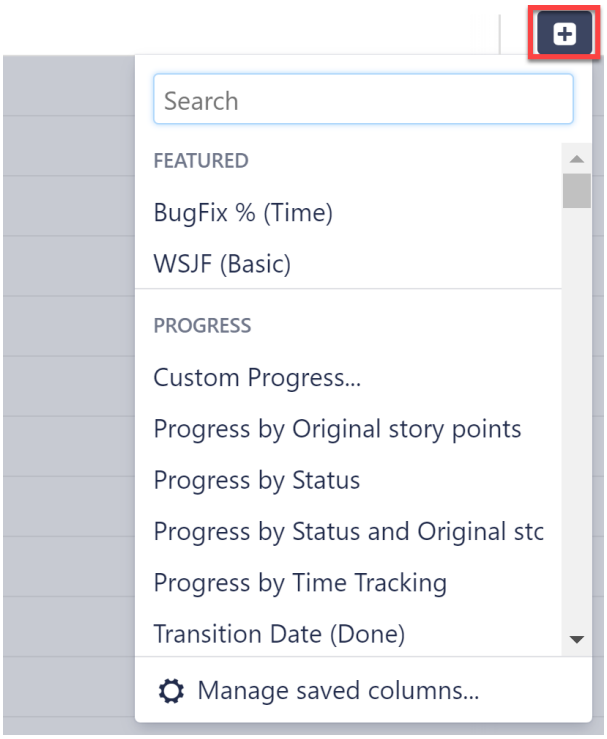
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add initiatives: **Automation | Insert | JQL Query |** enter the appropriate JQL, such as `"Project = 'project name' AND issuetype = initiative"`
3. Add epics below initiatives: **Automation | Extend | Linked Issues |** choose the type and direction used to assign Epics to Initiatives, such as "Implements" or "parent is Implemented by sub-issue"
4. Add issues belonging to epics: **Automation | Extend | Stories under Epics**
5. Add sub-tasks: **Automation | Extend | Sub-tasks**
6. Sort by rank: **Automation | Sort | type "Rank"**

4.2.5.6 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate

- Progress[130] - track status based on issue progress, percent completion, or resolution
- Tempo Work Logged[131] - track how long your team is spending on each issue
- Formulas[132] - create your own formulas to compare fields or create a visual report
- Time in Status[133] - see how much time issues spend in a particular status
- Last Comment[134] - view the latest comment for each issue



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

Take SAFe management to the next level

By adding Structure.Pages[135], you can manage your Strategy documentation and linked issues all in one place.

Learn More[136]

Advanced SAFe Planning

The following guide will walk you through an advanced SAFe implementation in Structure.

130 https://wiki.almworks.com/display/structure/Progress+Column
131 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
132 https://almworks.com/structure/help/column/formula
133 https://wiki.almworks.com/display/structure/Time+in+Status+Column
134 https://almworks.com/structure/help/column/last-comment
135 https://alm.works/pages
136 https://almworks.com/structure/help/how-to/safe

4.2.6 Requirements

To do this, you will need:

- Jira Software and Confluence
- Structure
- Structure.Pages
- Structure.Testy

Ready-to-use demo space: Live Demo[137]

4.2.7 Before You Begin

We assume that your Jira is prepared for scaling. The most common configuration is to create separate projects for each level:

- Kanban project for Portfolio level, with the following Epic issue type and statuses: Funnel, Reviewing, Analyzing, Portfolio Backlog, Implementing and Done
- Kanban project for Solution level, with the following Issue types: Capability and Enabler
- Kanban project for Program level, with the following issue types: Feature and Enabler

> ⚠ Features are named "Epics" in Jira, so you can rename them in Administration. Projects for each Team (Scrum or kanban, Scrum is preferable) and Issue types (User Story, Bug, Improvement, Enabler) can be left as is.

4.2.8 Step-by-step guide

1. **Define your company Strategy in Confluence**
   Create a Space in Confluence for high-level documents and describe the Strategy of your company.

2. **Define Themes**
   Define Strategy more precisely by dividing it into several themes, each a sub-page of Strategy.

3. **Link Themes to Epics**
   Define Epics, or high-level tasks for implementing your theme. Link Jira issues with their corresponding themes in Confluence.

4. **Add Capabilities**
   Since you have a separate project for Capabilities, divide your Epics into more precise Capability issues and put them into a Capability Board. Don't forget to link each Capability with a corresponding Epic! You can use standard Jira issue link types, but it will be more convenient to create you own[138] "implements"/ "is implemented by" link type.

5. **Add Features**
   Add details to any large Capabilities and divide them into Features. Don't forget to link each Feature to its corresponding Capability.

---

137 http://alm.works/safe-demo
138 https://confluence.atlassian.com/adminjiraserver071/configuring-issue-linking-802593137.html

6. **Stories**
   Each Feature can be divided into more detailed Stories. Don't forget to link Features and Stories with an epic link.

7. **Sub-tasks**
   If you need more detailed tasks, divide Stories into Sub-tasks.

8. **Put it all together with Structure**
   Create an empty Structure and visualize everything in one place: http://recordit.co/lFYAzeBezV
   - Find your Strategy page and add it to the Structure: http://recordit.co/0vt6mNYsqR
   - Click the Automation button to enable automation. Then click the **+** icon next to the **Automation** button, and use the Extend generator to add the following rules:
     - Add all child pages (extend | child pages): http://recordit.co/5l5fqPM4c3
     - Add issues linked to pages
     - Add Capabilities linked to Epics
     - Add Features linked to Capabilities
     - Add Stories linked to Features
     - Add add Sub Tasks linked to Stories

9. **Take it to the next level!**
   Now that your structure is created, you can modify it to fit your needs:
   - You can Group issues on a specific level. For example, group by Sprint on the Team level to track progresses.
   - You can prioritize issues by sorting by WSJF.
   - You can specify your Objective during PI planning in Notes.

4.3 Project Manager

Custom Reports

With Structure, you can easily create a visual overview of your projects – and visualize all the data you need to track on a single screen.

4.3.1 Step 1: Build Your Structure

To create a new structure:

- Go to the Jira menu and select **Structure | Create Structure**
- Add the issues you want to track – you can add them manually or automatically using **Automation | Insert**
- Add related issues - you can do this using **Automation | Group** and **Automation | Extend**

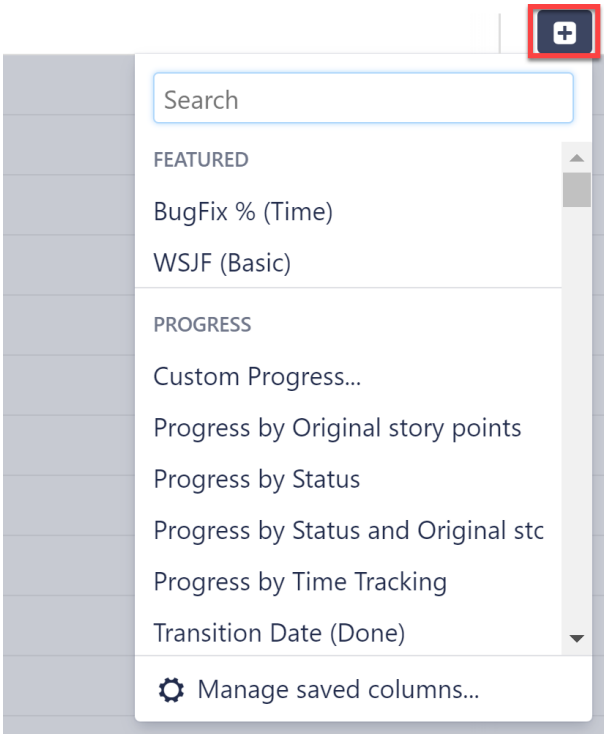
4.3.2 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate
- Progress[139] - track status based on issue progress, percent completion, or resolution

139 https://wiki.almworks.com/display/structure/Progress+Column

- Tempo Work Logged[140] - track how long your team is spending on each issue
- Formulas[141] - create your own formulas to compare fields or create a visual report
- Time in Status[142] - see how much time issues spend in a particular status
- Last Comment[143] - view the latest comment for each issue



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

Manage Epics and Tasks

With Structure, you can easily organize issues into an advanced Agile hierarchy (Epics > Issues > Sub-tasks) AND identify any issues that aren't attached to epics.

4.3.3 Step 1: Create a New Structure

1. Go to the Jira menu and select **Structure | Create Structure**
2. On the Template page, select **Empty Structure**

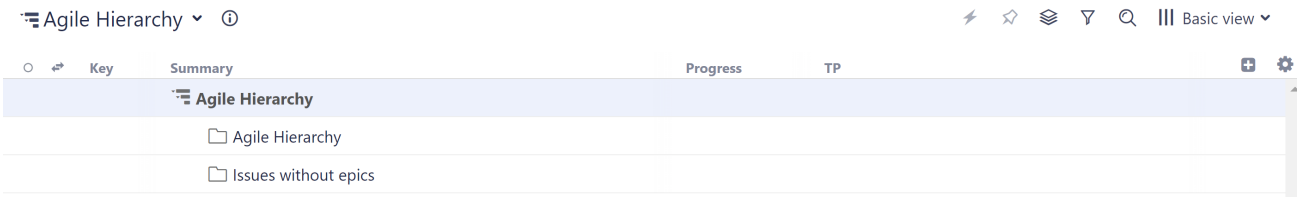
4.3.4 Step 2: Add Two Folders

Open the **+Add** menu and select **Folder**. Add two folders, named:

- "Agile Hierarchy"

140 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
141 https://almworks.com/structure/help/column/formula
142 https://wiki.almworks.com/display/structure/Time+in+Status+Column
143 https://almworks.com/structure/help/column/last-comment

- "Issues without Epics"

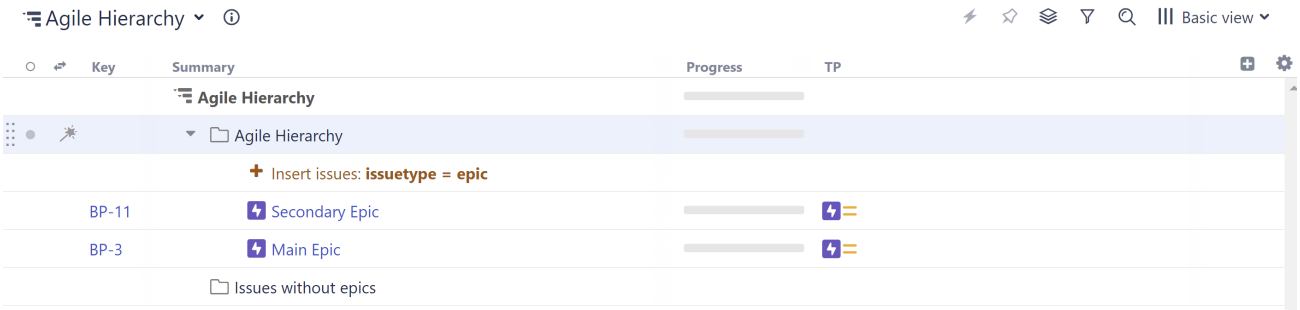


4.3.5 Step 3: Insert Epics

Select the Agile Hierarchy folder, so that its row is highlighted, and add a JQL Query Insert Generator:

1. Go to **Automation | Insert | JQL Query**
2. Enter the following JQL query: `issuetype = Epic`
    - To limit the epics to specific projects or other variables, add additional specifications (Example: `AND project = "My Epics"`)
3. Click **Create**

All your epics should now be placed beneath the Agile Hierarchy folder.

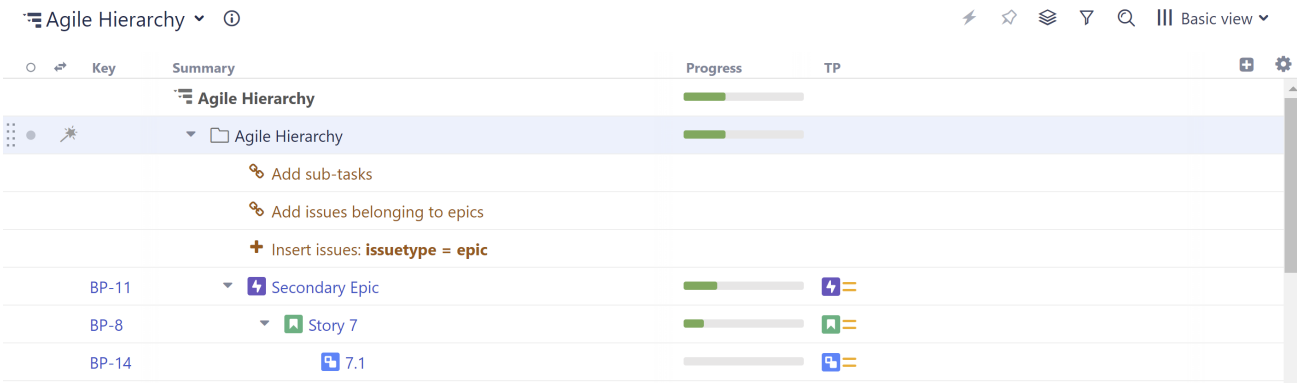


If the epics did not appear beneath the Agile Hierarchy folder, your generator was probably placed at the top of the structure, instead of beneath the folder. Select the "Insert issues: issuetype = Epic" row and drag it to beneath the Agile Hierarchy folder (see the picture above).

4.3.6 Step 4: Insert Issues and Sub-tasks

Make sure the Agile Hierarchy folder is still selected, return to the **Automation** menu, and add the following two generators:

- **Extend | Stories under Epics...**
- **Extend | Sub-tasks...**

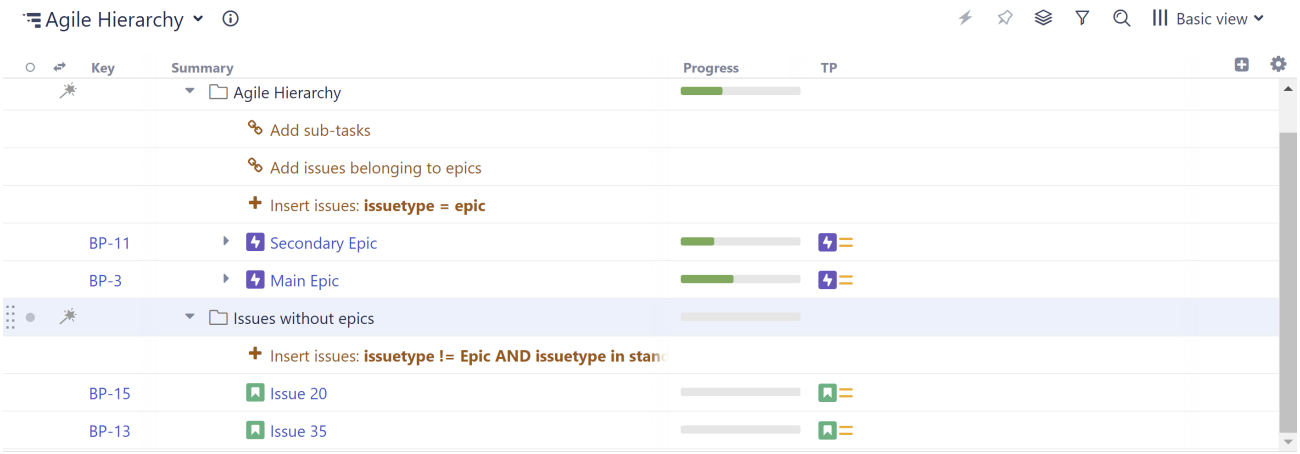


4.3.7 Step 5: Insert Issues without Epics

Next, we need to make sure we don't lose track of any issues that aren't linked to epics yet. That's what our second folder is for:

1. Select the "Issues without Epics" folder
2. Go to **Automation | Insert | JQL Query**
3. Enter the following JQL query: `issuetype != Epic AND issuetype in standardIssueTypes() AND "Epic Link" is empty`
    - If you narrowed your query to specific projects, etc. for your epics, do the same here
4. Click **Create**

Any issues that are not assigned to epics should now appear beneath the second folder.



If no issues appear under the Issues without Epics folder, make sure the Insert Generator is located in the right location - if not, drag the generator to beneath the Issues without Epics folder (see the picture above).

4.3.8 Step 6: Assign Issues to Epics / Reassign Issues

Now that you have all your issues organized and in one place, you can assign issues from the second folder to epics, or even reassign issues that already have epics.

4.3.8.1 Assigning Issues to Epics

To assign issues in the second folder to an epic:

- Drag the issues to the epic you want to assigned them to.
- While dragging, hold down the **ctrl** key - this copies the issue to the new location in the structure. Once the issue is copied to the new location, it should disappear from beneath the second folder (because now it's assigned to an epic).

Don't worry: copying an item in Structure does not make a copy in Jira.

4.3.8.2 Reassigning Issues to New Epics

To reassign issues from one epic to another, simply drag them from beneath their current epic to the new epic.

Release Management

With Structure, you can track all the issues for upcoming releases – even if they're from different projects – and visualize all the data you need to track on a single screen.

4.3.9 Step 1: Build a Release Management Structure

To create a structure for release management:

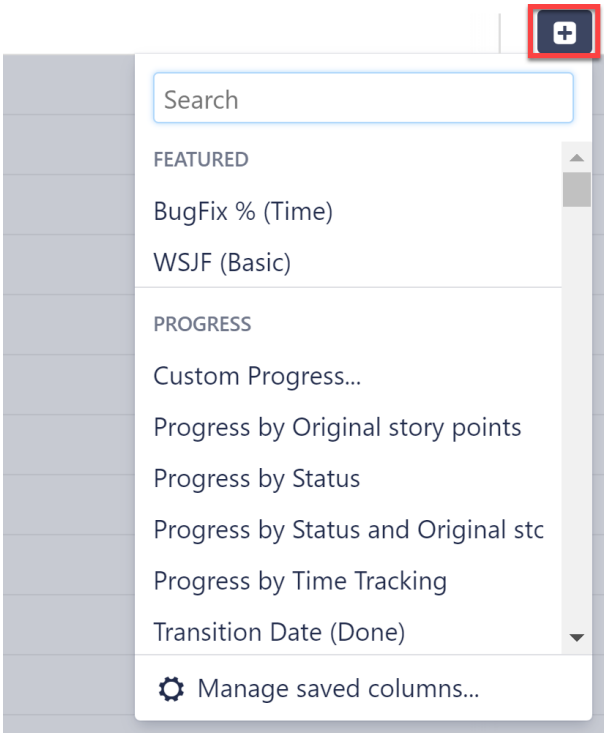
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add the issues you want to track – you can add them manually or automatically using **Automation | Insert**
3. Group issues by fix version: **Automation | Group | type "Fix Version/s"** *(Tip: If you use the same names in different projects, use **"Version name..."** instead.)*
4. Open the Summary settings and select **Show Sprint and Version attributes**

4.3.10 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate
- Progress[144] - track status based on issue progress, percent completion, or resolution
- Tempo Work Logged[145] - track how long your team is spending on each issue
- Formulas[146] - create your own formulas to compare fields or create a visual report
- Time in Status[147] - see how much time issues spend in a particular status
- Last Comment[148] - view the latest comment for each issue

144 https://wiki.almworks.com/display/structure/Progress+Column
145 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
146 https://almworks.com/structure/help/column/formula
147 https://wiki.almworks.com/display/structure/Time+in+Status+Column
148 https://almworks.com/structure/help/column/last-comment



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

Synchronize Links Across Teams or Projects

When you're working across multiple projects and teams, chances are there are some inconsistencies between the link types being used.

The following guide will show you how to quickly update all your links to a single, consistent link type using Structure.

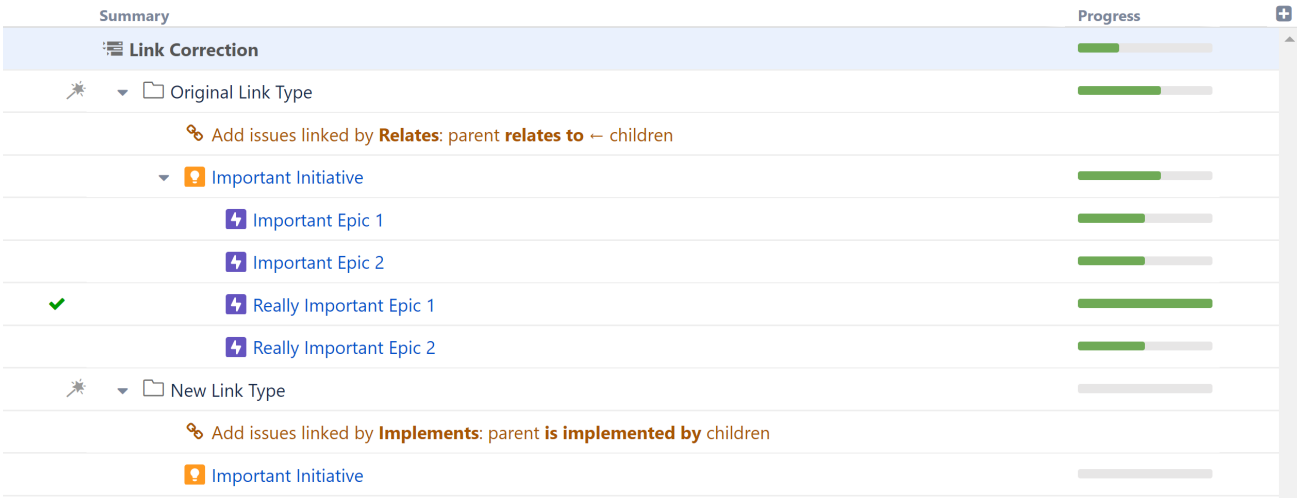
> The examples below demonstrate how to update link types for an initiative, but the same process can be used for other link types as well.

4.3.11 Step 1: Build a Simple Structure

1. Create a new, blank structure and add 2 folders: "Original Link Type" and "New Link Type".
2. Manually add your parent initiative to both folders. *Note: this should be the same initiative for both.*

4.3.12 Step 2: Add Epics

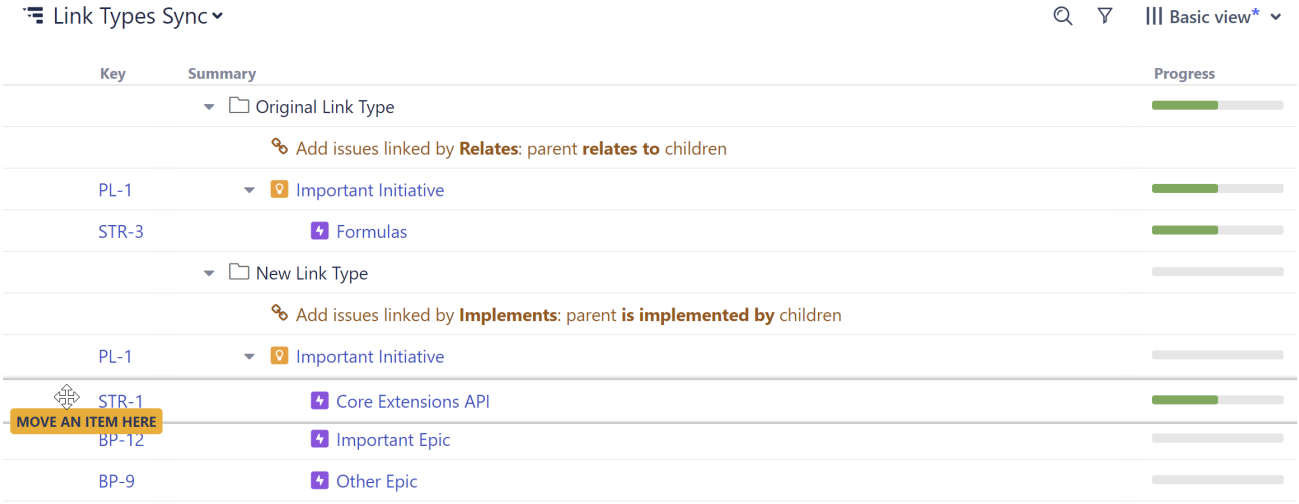
1. Highlight the Original Link Type folder and add a Linked Issue generator: **Automation | Extend | Linked Issues |** select the current (inconsistent) link type
2. Highlight the New Link Type folder and add a Linked Issue generator: **Automation | Extend | Linked Issues |** select the desired link type



> Make sure the **Allow changes via Structure** option is checked for both generators.

4.3.13 Step 3: Update Link Types

Select all the issues under the generator added to the first folder, and **drag them to the second folder**, as children of the same initiative.



Structure will automatically replace all the old links with the new link type.

Need to keep the original links too? When dragging the items, hold down the "ctrl" key (Option key on Mac) to copy the items to the new iniative.

4.3.14 Additional Use Cases

You can also use this same method to:

- Update other link types - just edit your Extend | Link generators and the original item added to each folder
- Remove links - include an empty folder (no Extend generator) and drag your linked issues to the empty folder - the original link will be removed without adding a new link

Track Overdue Tasks

Use Structure to identify:

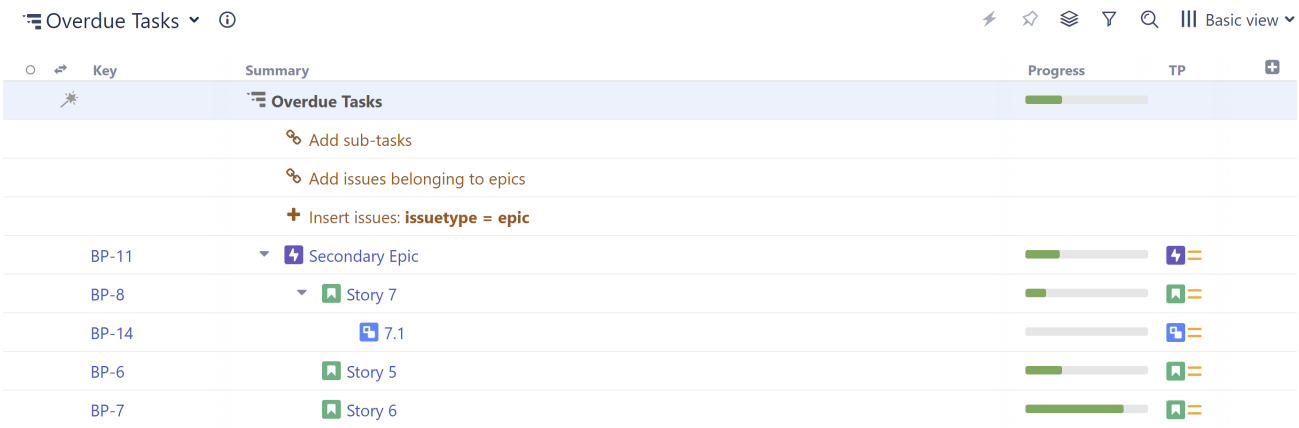
1. Overdue tasks
2. Number of overdue tasks per epic, initiative, or other larger grouping
3. Ratio of overdue tasks to total tasks for each epic, initiative, or other large grouping

4.3.15 Step 1: Build Your Structure

In this guide, we're going to show you how to track overdue tasks at the epic level. To track overdue tasks for initiatives or any other large groupings, simply reconfigure the structure hierarchy - this can be done using different Insert(see page 142), Extend(see page 148), and Group (see page 169)generators.

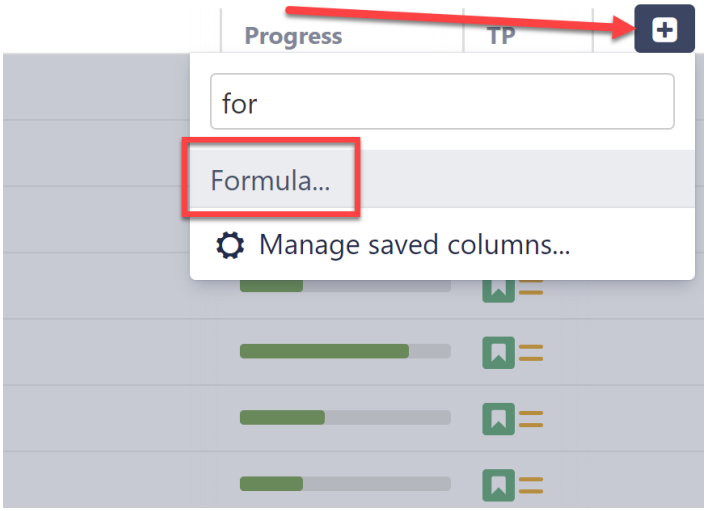
To create a new structure:

1. Go to the Jira menu and select **Structure | Create Structure**
2. Add epics: **Automation | Insert | JQL Query** and enter the following JQL query: `issuetype = Epic`
    - To limit the epics to specific projects or other variables, add additional specifications (Example: `AND project = "My Epics"`)
3. Add stories: **Automation | Extend | Stories under Epics...**
4. Add sub-tasks (optional): **Automation | Extend | Sub-tasks...**

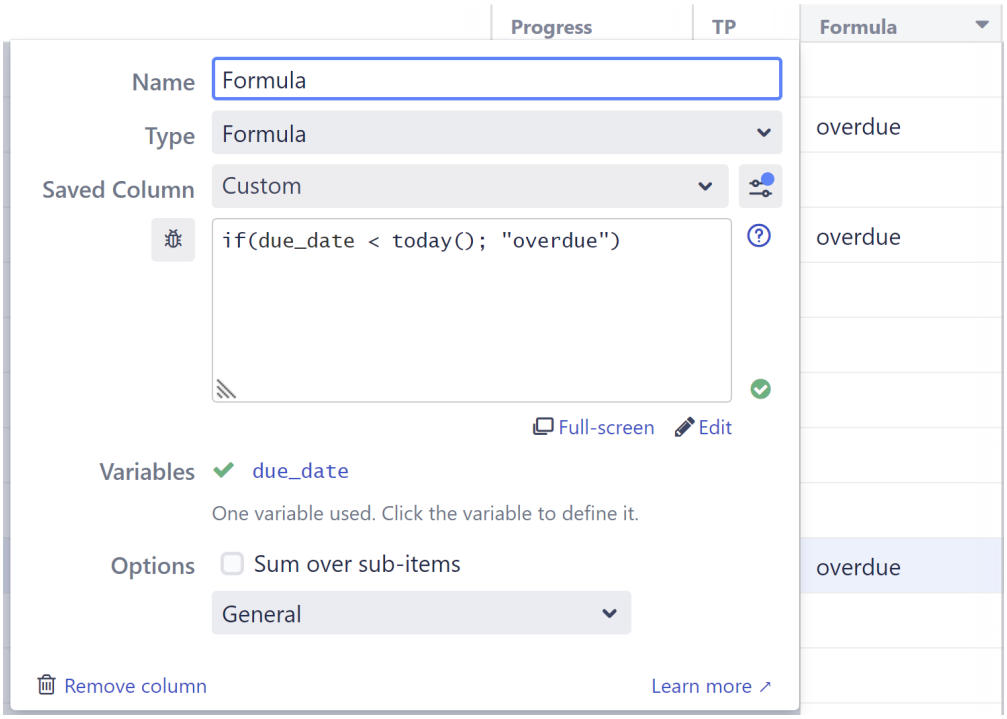


4.3.16 Step 2: Mark Overdue Tasks

Once you have your structure in place, the next step is to identify all the overdue items. To do this, we need to add a formula column(see page 625). Click the **+** button to the right of the column headers, and select **Formula.**



Give the column an appropriate name and enter the following into the Formula section: `IF(due_date`[149] `< today(); "overdue")`



149 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6

If you used "due_date" in your formula, the variable will automatically be mapped to the Due Date attribute. If you used another name for that variable (for example, “due”), you will need to define the variable by pointing it to the appropriate Due Date attribute.

4.3.16.1 Variations

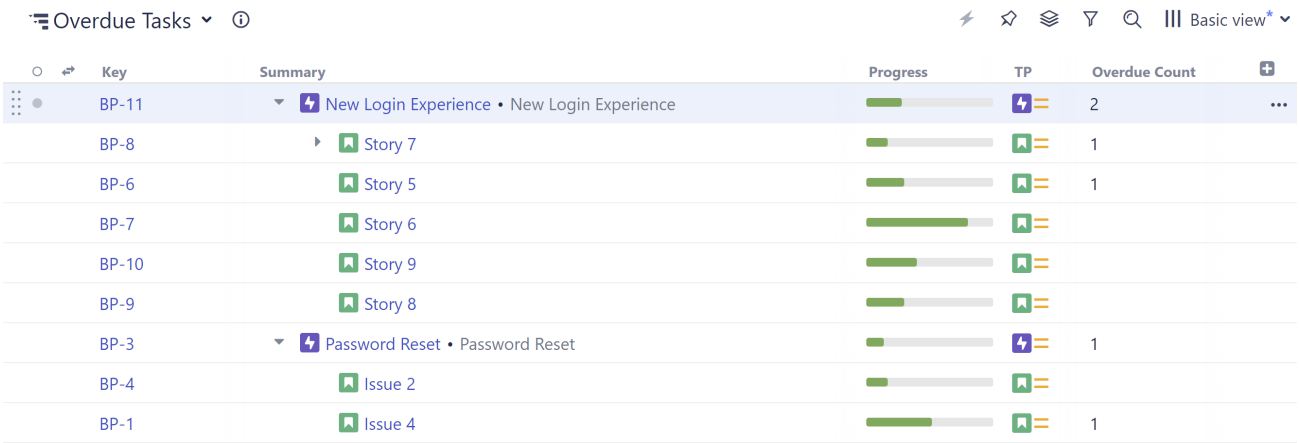
This is just one way to mark overdue items. You could also:

- Create a more visual warning that highlights not only overdue items, but also items coming due. You can read how to do this in Wiki Markup in Formula Columns(see page 251).
- Create a formula that relies on another due date. For example, you may want to flag issues that aren't completed by the due date of their parent epic. In this case, change the formula to: `IF(type[150] = "Story" and status[151] != "Done" and parent{due_date[152]} < today(); "overdue")`
- Aggregate overdue items for epics (we'll cover this next!)

4.3.17 Step 3: Aggregate Overdue Items for Epics, Initiatives, etc.

In the previous example we were simply returning the string “overdue”, but if we give that "overdue" flag a numeric value ("1"), we can use the Sum over sub-items option to aggregate those values up the hierarchy. To accomplish this:

1. In place of the formula we created in step 1, enter the following: `IF(type[153] = "Story" and status[154] != "Done" and due_date[155] < today(); 1)`
2. Check the Sum over sub-items box



In this example, we limited our scope to only stories that are not yet "Done" - depending on your situation, you may want to adjust this to include other issue types, add additional qualifications, etc.

150 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
151 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
152 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
153 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
154 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
155 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6

4.3.18 Step 4: Calculate the Percentage of Stories That Are Overdue

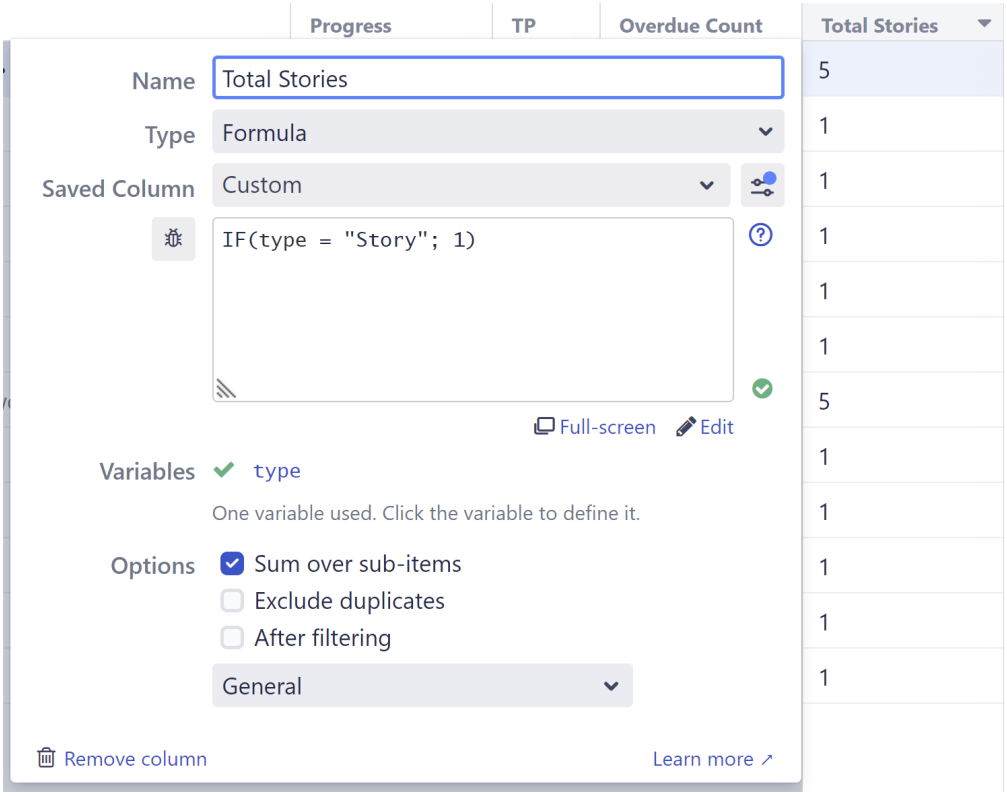
So far, we've just identified and counted our overdue stories. But this doesn't paint a complete picture. And epic with 1 overdue item out of 257 is very different from 1 out of 5!

To get a better understanding of how these overdue items relate to the big picture, next we're going to:

- Calculate total stories for each higher level in our hierarchy
- Calculate the percentage of stories that are overdue for each higher level

First, we need to create another Formula column, which we'll use to calculate our total stories. For this column:

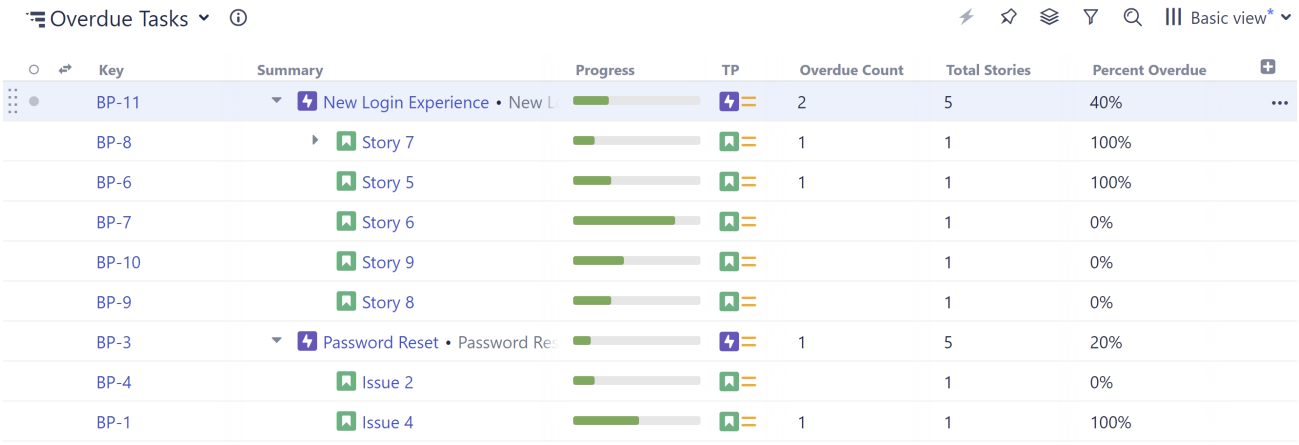
- Enter the following formula: `IF(type = "Story"; 1)`
- Check the Sum over sub-item box



Next, we'll create a third formula column to compare our values from the first two. For this column:

1. Enter the formula: `IF(total_stories`[156]`; overdue`[157]` / total_stories`[158]`)`
2. Map your "total_stories" and "overdue" variables to the appropriate columns we created above

156 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
157 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
158 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6



For this formula, we wrapped the expression in an if statement so we wouldn't get a DIV/0 error on lines without a value for total stories.

At this point, you can keep all three columns in your structure (we recommend saving them as a new view(see page 511)), or get rid of one or both of your original columns - our Percent column will keep their original calculations, even if those columns are deleted or changed. (This also means if you make changes to the formulas used in the Overdue or Total Stories column, you'll need to re-select those columns in the Variables section of your Percent formula to apply those changes there.)

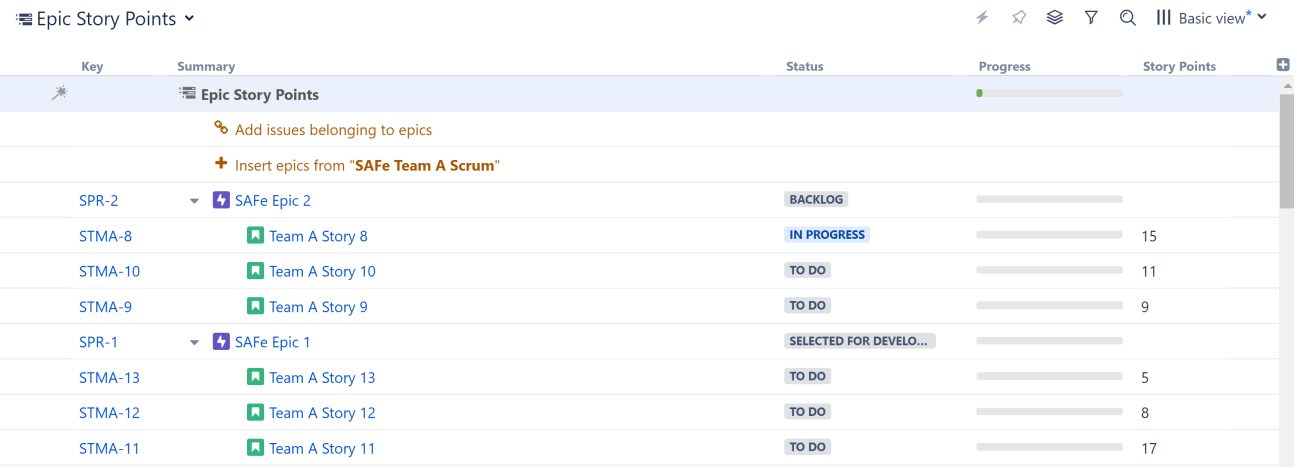
5 Solve Common Tasks with Structure

Learn to build custom structures to accomplish your daily tasks.

5.1 Calculate Epic Story Points Based on Sub-issues

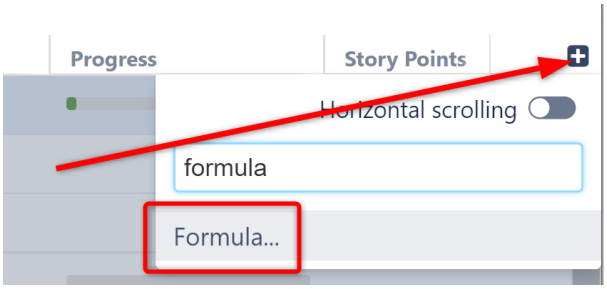
The following article will show you how to update the Story Point value for each epic based on the combined value of all the stories/tasks within the epic.

- We will use a Formula column(see page 637) to calculate the total story points for each epic, and then use an Effector (see page 179)to update the epic's Story Point field.
- In this example, we've used Insert(see page 142) and Extend(see page 148) generators to build an Epic > Story hierarchy. Notice that our Epics do not have Story Point values, but the stories assigned to them do.

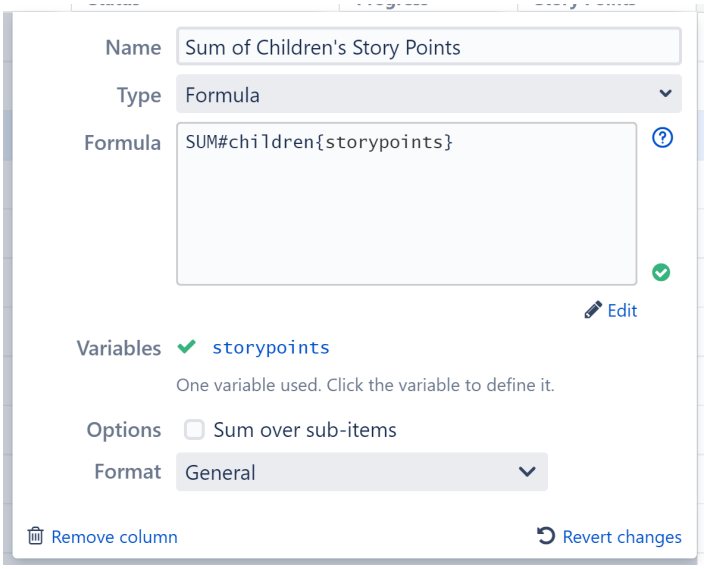


5.1.1 Step 1: Add a Formula Column to Calculated the Total Story Points

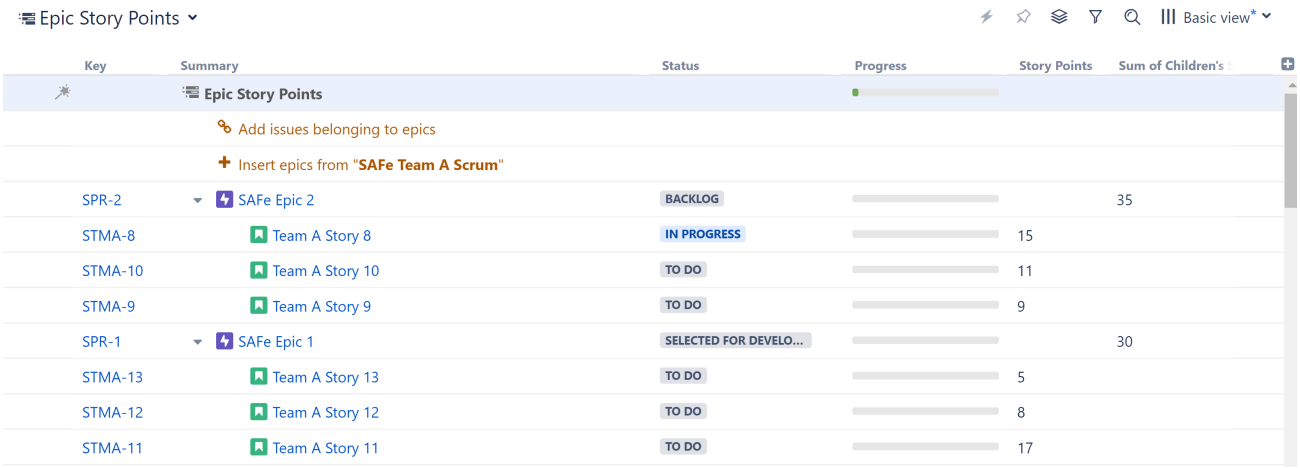
Click the **+** icon to the right of the column headers, and select **Formula...**



To calculate the total story points for each epic, we're going to use the aggregate function(see page 387) "SUM" with the "#children" modifier. Give the column an appropriate name and enter the following formula: `SUM#children{storypoints}`



Since we only have epics and stories in our structure, this will only create a value for our epics, because our stories don't have any children.



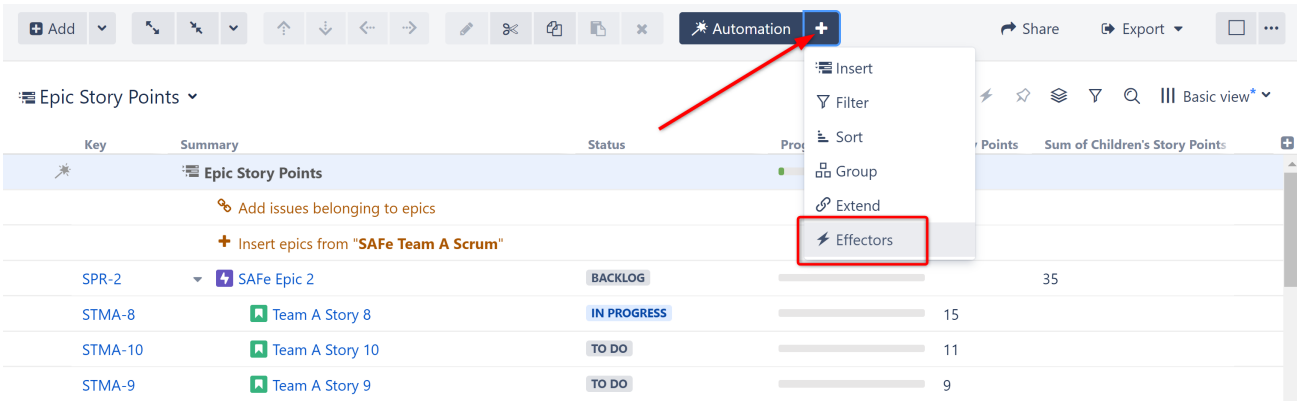
These aggregate values are only available in Structure, but we can use an Effector to take our Epic totals and write them to the Story Points field.

If you have an additional level below stories (such as sub-tasks), you will see Sum values for your stories as well - you can disregard those, because we're going to tell the Effector to ignore them when updating our Story Point values.

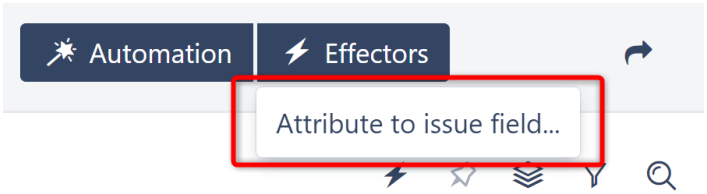
5.1.2 Step 2: Add an Effector

Now we need to take the information from the formula column and write it to the Story Points field for each Epic.

Open the Automation menu and select **Effectors**.

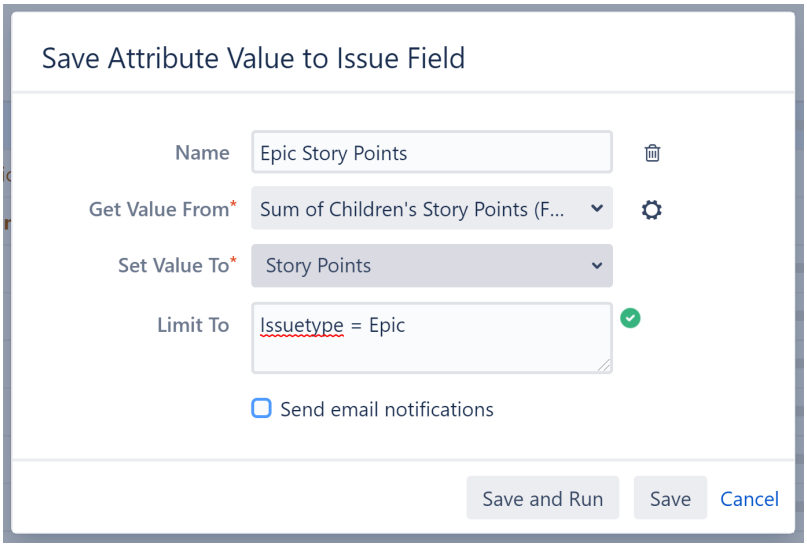


Choose **Attribute to issue field...**



On the Effector setting screen:

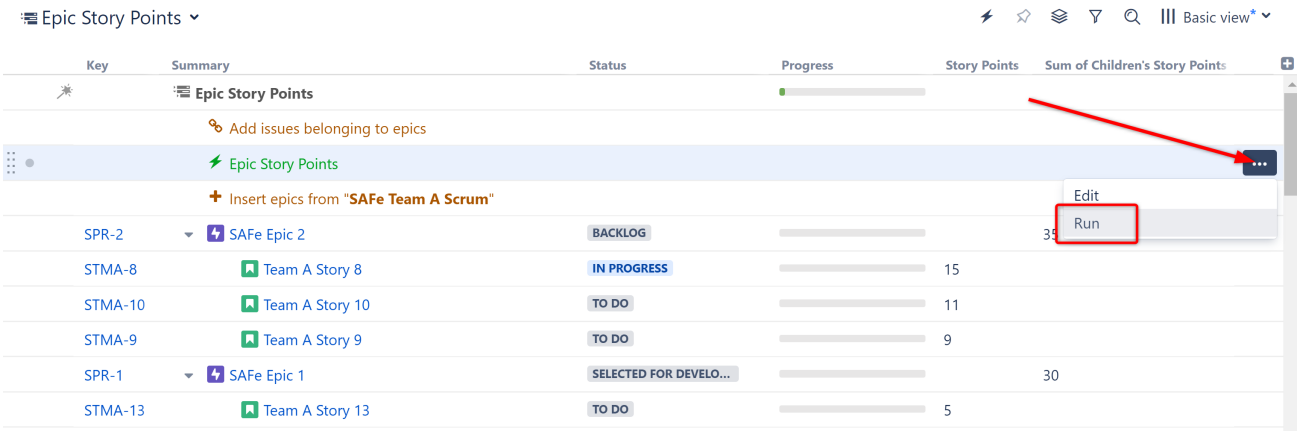
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Get Value From: Select the formula column we added in Step 1
- Set Value To: Story Points
- Limit To: Since we only want to add Story Point values for our Epics, add the following JQL Filter: Issuetype = Epic (*Note: This is very important! If you don't set this filter, the Story Points data for stories will be overwritten by their children's sums, or removed completely if they have no children.*)
- Select whether email notifications should be sent when the Effector writes values to Jira



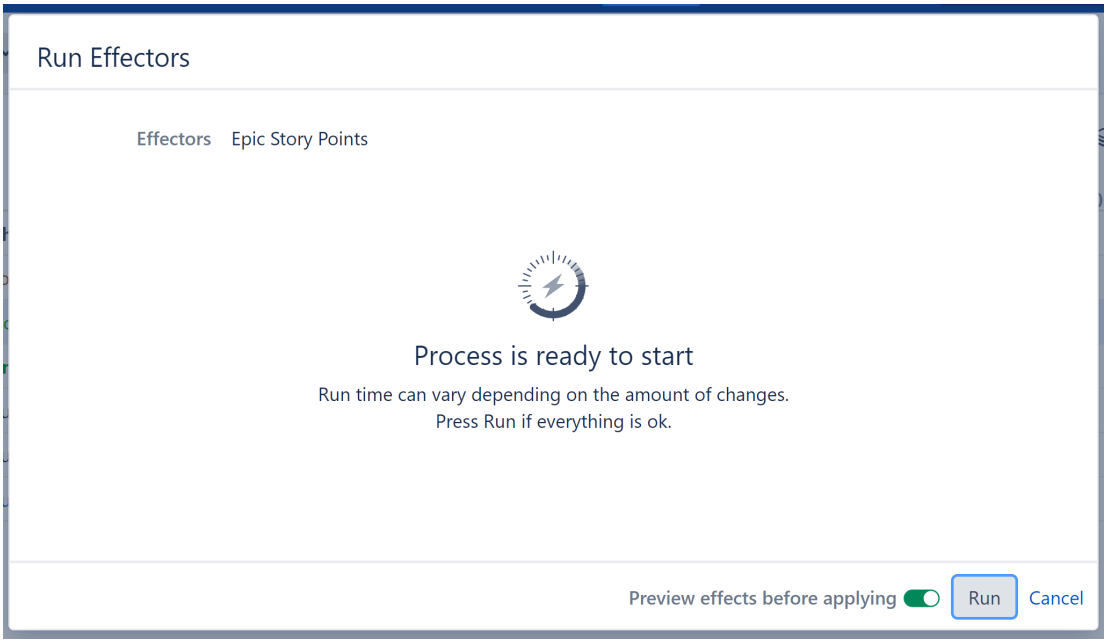
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

5.1.3 Step 3: Run the Effector

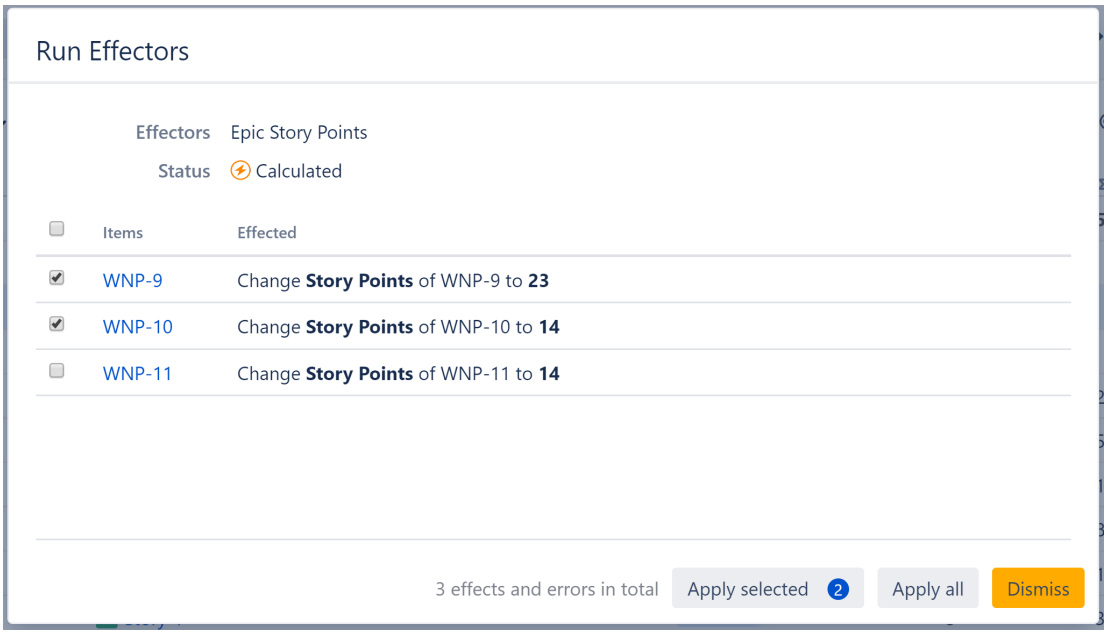
You can run an Effector directly from its settings screen (see above), or you can locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.



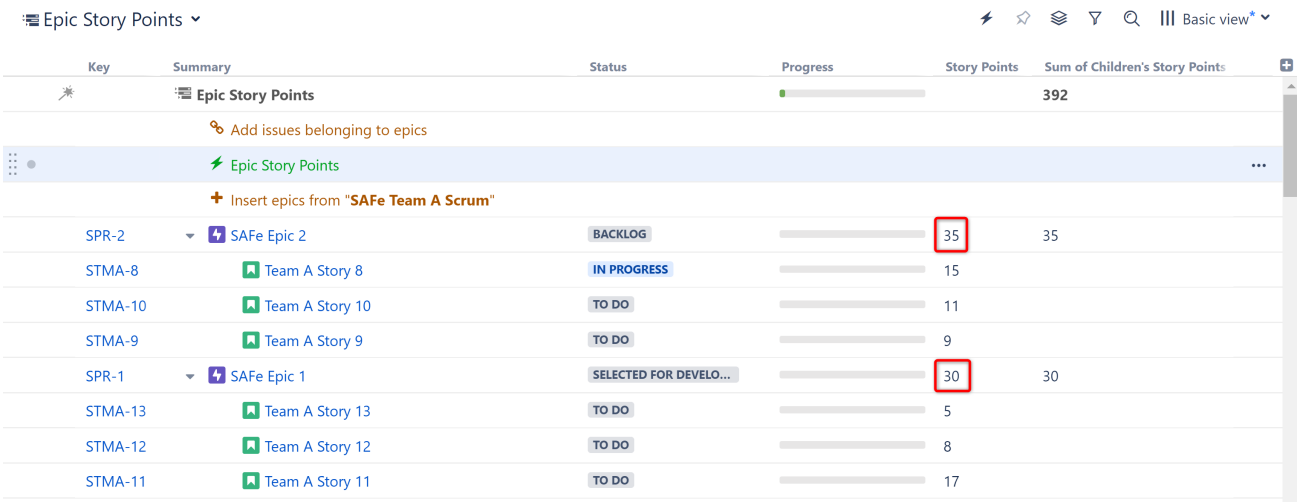
You have the option to **Preview effects before applying**. This allows you to view and approve every change that will be made by the Effector. Effectors update live Jira data, so we highly recommend using the preview option. Click **Run** to begin.



Once the preview is finished, you will see a list of changes that will be made by the Effector. You can select which changes you want made, or click **Apply All** to apply all changes.



Once you apply the selected changes, the calculated Story Points value will be written to each of our Epics:



5.2 Cost Calculation for Issues and Projects

Design a structure for calculating what it will cost to complete each Story, Epic, or other grouping of issues, based on the amount of time remaining and the hourly rate of the assignee.

5.2.1 Requirements

For this method to be successful, you need:

- All stories and tasks assigned to specific Jira users
- A custom User property[159] containing each user's hourly rate of pay (we've named ours "cost")

5.2.2 Step 1: Build a Hierarchy

Build a structure with an Epic > Story > Sub-task hierarchy:

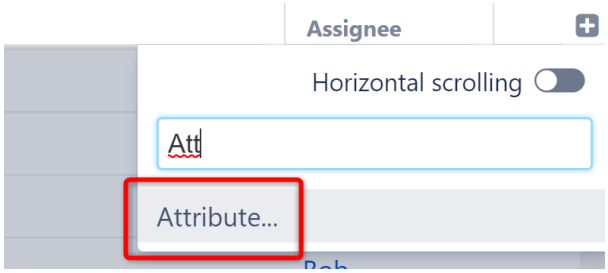
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add epics: **Automation | Insert | JQL Query** and enter the following JQL query: `issuetype = Epic`
    a. To limit the epics to specific projects or other variables, add additional specifications (Example: `AND project = "My Epics"`)
3. Add stories: **Automation | Extend | Stories under Epics...**
4. Add sub-tasks (optional): **Automation | Extend | Sub-tasks...**

There are many ways to build your hierarchy, depending on what values you want to track. For more ideas, check out some of our other Task Articles(see page 637) or our Generators(see page 140) docs.

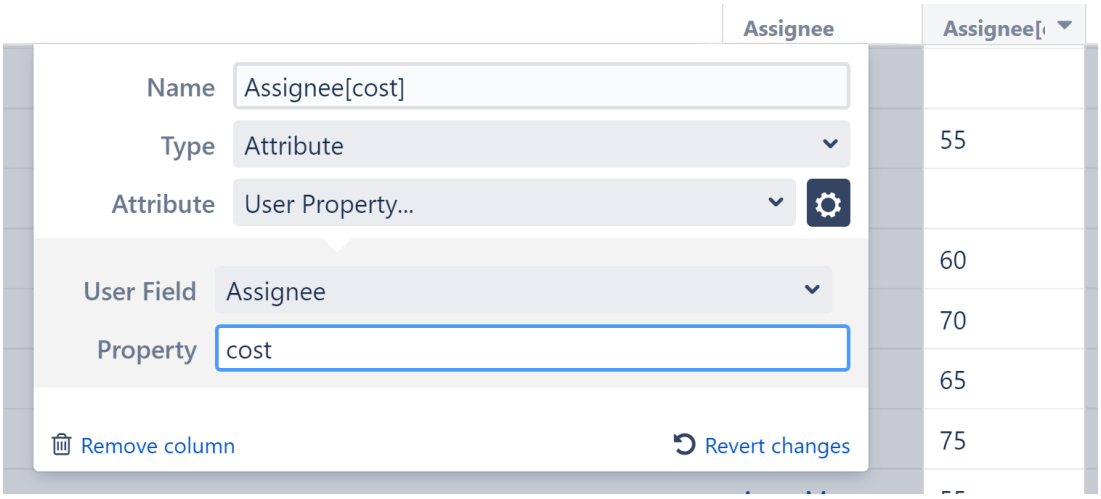
159 https://confluence.atlassian.com/adminjiraserver/create-edit-or-remove-a-user-938847025.html

5.2.3 Step 2: Add a Column for the Custom User Property

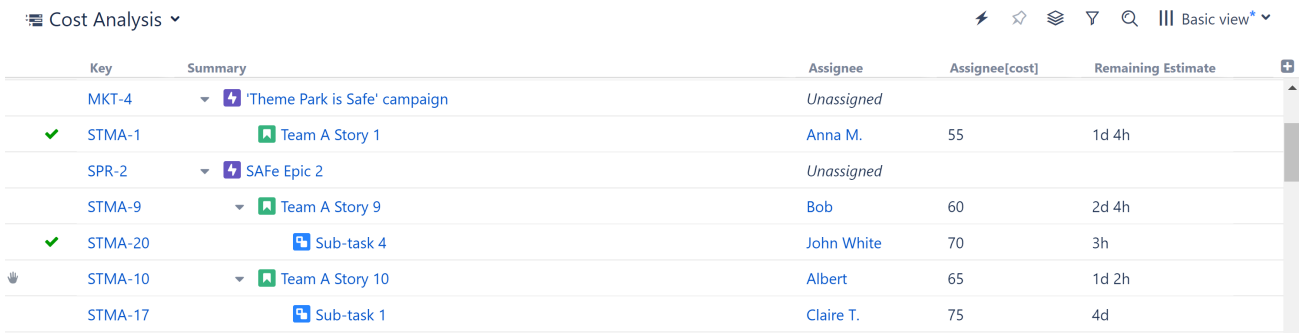
1. Open the Add Column menu (**+**) and select **Attribute...**



2. Under the column properties, enter the following values:
   - **Attribute:** User Property...
   - **User Field:** Assignee
   - **Property:** Enter the key for the user property where you store each user's hourly pay rate - ours is called "cost"



You should now have an **Assignee(cost)** column that displays the values from our custom user property:

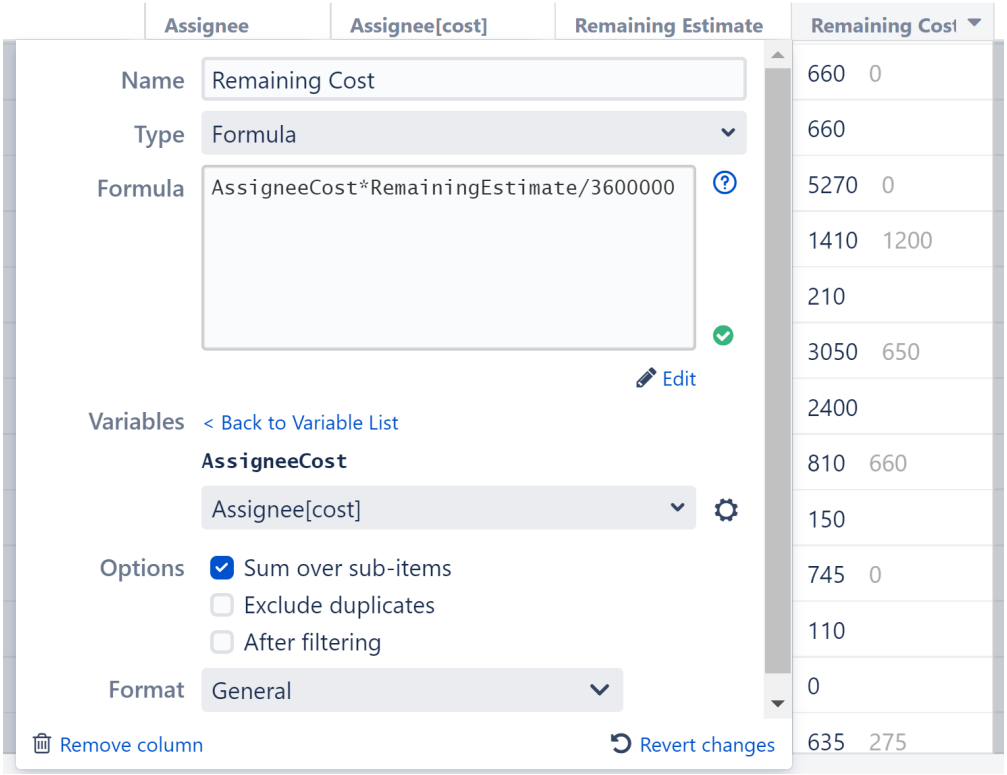


Key	Summary	Assignee	Assignee[cost]	Remaining Estimate
MKT-4	'Theme Park is Safe' campaign	*Unassigned*		
STMA-1	Team A Story 1	Anna M.	55	1d 4h
SPR-2	SAFe Epic 2	*Unassigned*		
STMA-9	Team A Story 9	Bob	60	2d 4h
STMA-20	Sub-task 4	John White	70	3h
STMA-10	Team A Story 10	Albert	65	1d 2h
STMA-17	Sub-task 1	Claire T.	75	4d

5.2.4 Step 3: Calculate the Cost to Finish Each Issue (epic/story/etc.)

Add a formula column(see page 642) to calculate the cost to finish each issue in our structure. Enter the following:

- Formula: `AssigneeCost*RemainingEstimate/3600000`
- Variables: map AssigneeCost to the custom "cost" property we discussed in Step 2 - see Columns as Variables(see page 233) for more details
- Check **Sum over sub-items** so the cost is aggregated up the hierarchy
- If issues or tasks appear multiple times within an epic (or other grouping), check **Exclude deplicates** to avoid an inflated total costs



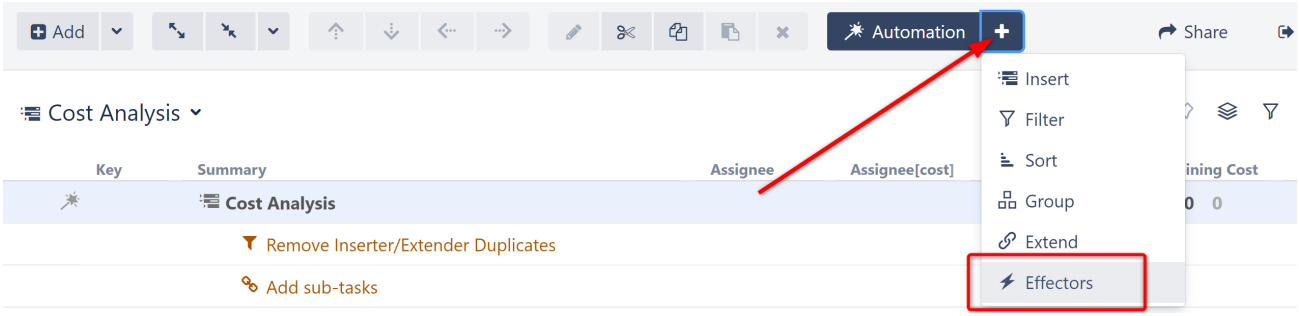
5.2.5 Bonus - Make This Information Visible Outside of Structure

Using Effectors(see page 179), we can write the values from our formula to a custom Jira field, making them visible to users outside of Structure.

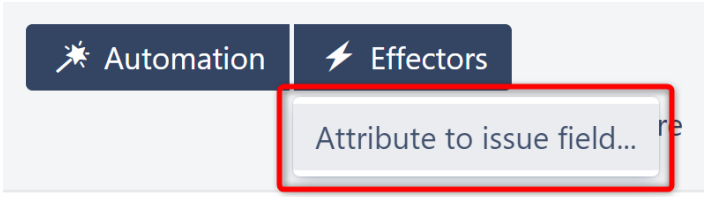
Before we start, you'll need to make sure you have a custom field where the Cost-to-Complete values can be stored. We created a custom field called "FinishCost".

5.2.5.1 Add an Effector

Open the Automation menu and select **Effectors**.

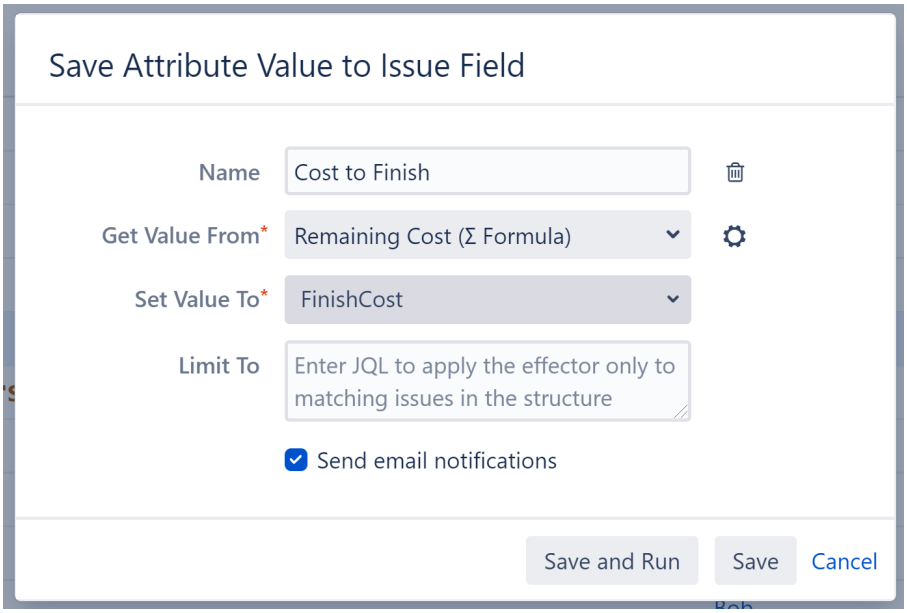


Choose **Attribute to issue field...**



On the Effector settings screen:

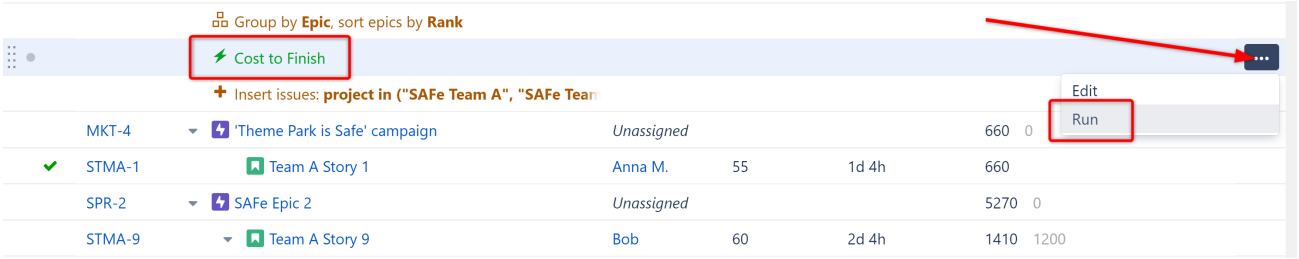
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Get Value From: select the formula column we added in Step 2
- Set Value To: select the custom Jira field you want to write the cost-to-complete values to
- Limit To: If you only want to write cost data for certain issues or issue types (for example, just for epics), enter the appropriate JQL. We've left this section blank, because we want to write values for all issues.
- Select whether email notifications should be sent when the Effector writes values to Jira



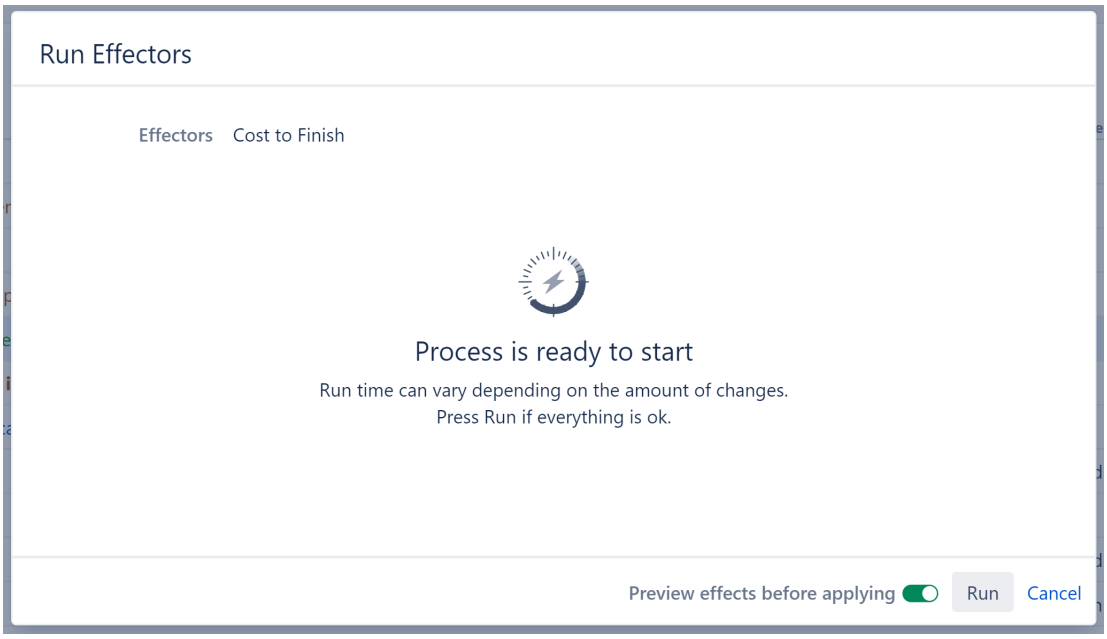
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

5.2.5.2 Run the Effector

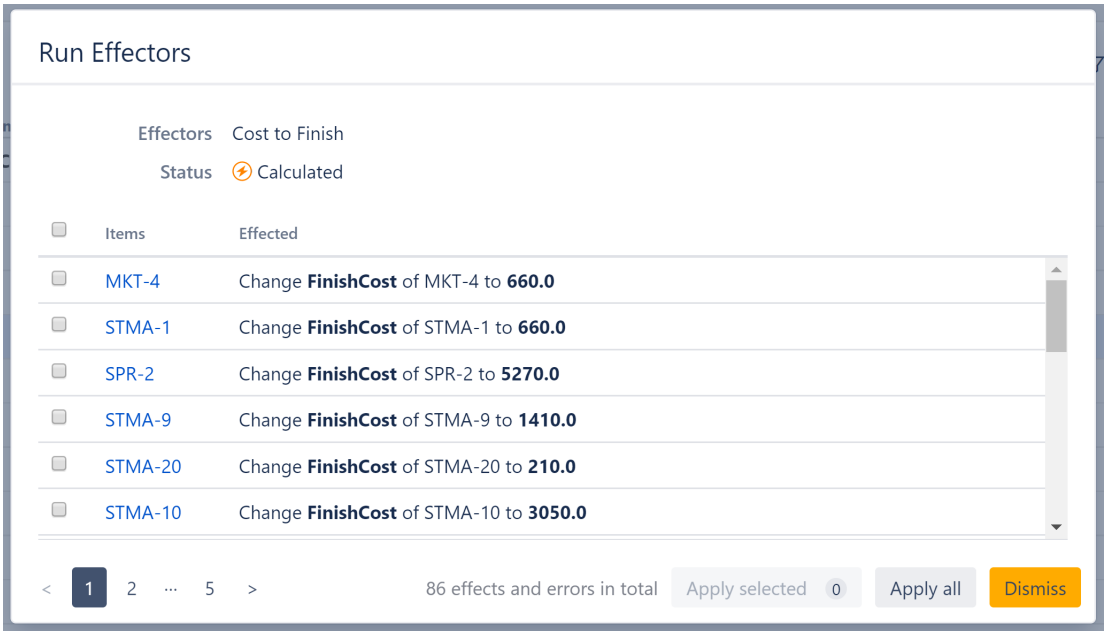
You can run an Effector directly from its settings screen (see above), or you can locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.



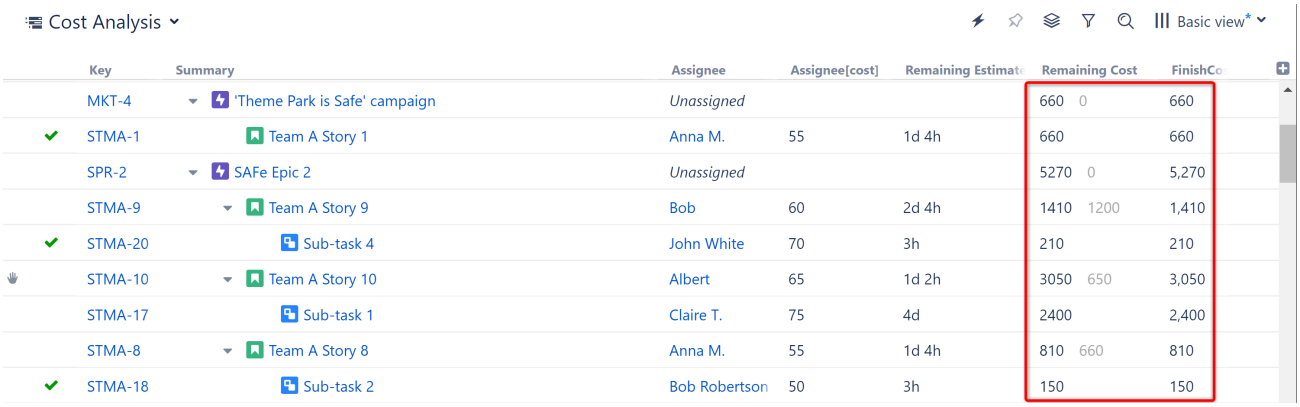
You have the option to **Preview effects before applying**. This allows you to view and approve every change that will be made by the Effector. Effectors update live Jira data, so we highly recommend using the preview option. Click **Run** to begin.



Once the preview is finished, you will see a list of changes that will be made by the Effector. You can select which changes you want made, or click **Apply All** to apply all changes.



Once you apply the selected changes, the cost values from your formula column will be written to the custom Jira field, where they can be viewed outside of Structure.



5.3 Manage Epics and Stories

With Structure, you can easily organize issues into an advanced Agile hierarchy (Epics > Issues > Sub-tasks) AND identify any issues that aren't attached to epics.

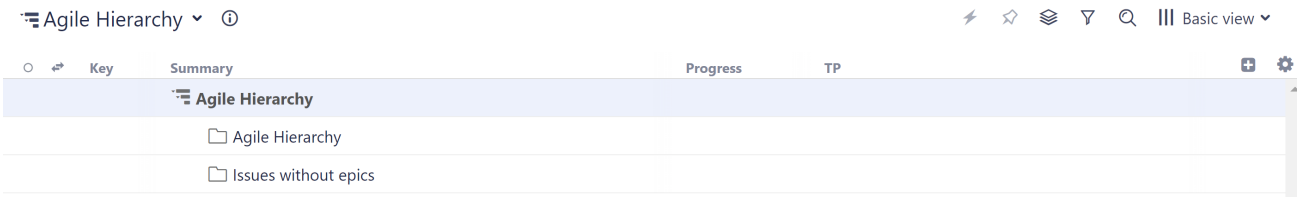
5.3.1 Step 1: Create a New Structure

1. Go to the Jira menu and select **Structure | Create Structure**
2. On the Template page, select **Empty Structure**

5.3.2 Step 2: Add Two Folders

Open the **+Add** menu and select **Folder**. Add two folders, named:

- "Agile Hierarchy"
- "Issues without Epics"

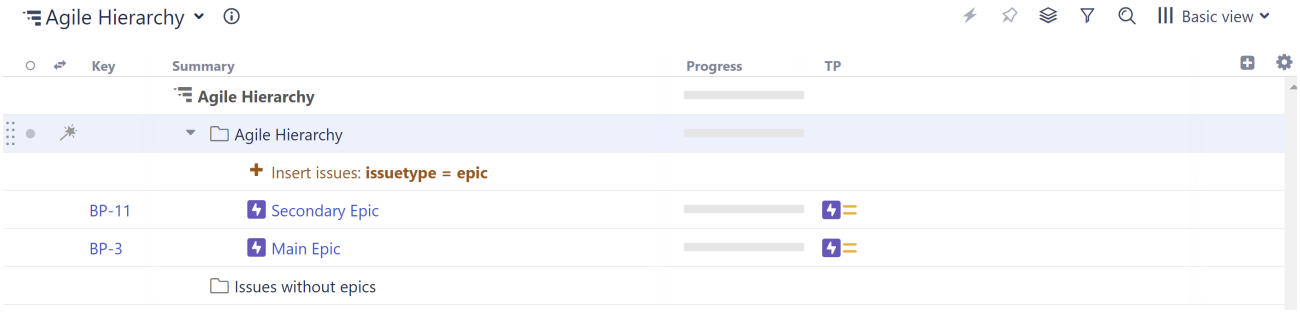


5.3.3 Step 3: Insert Epics

Select the Agile Hierarchy folder, so that its row is highlighted, and add a JQL Query Insert Generator:

1. Go to **Automation | Insert | JQL Query**
2. Enter the following JQL query: `issuetype = Epic`
    - To limit the epics to specific projects or other variables, add additional specifications (Example: `AND project = "My Epics"`)
3. Click **Create**

All your epics should now be placed beneath the Agile Hierarchy folder.

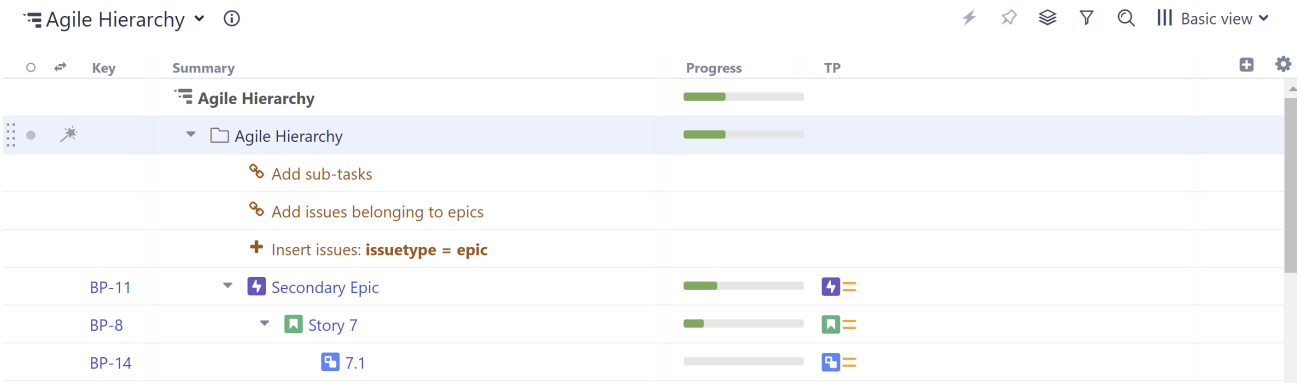


If the epics did not appear beneath the Agile Hierarchy folder, your generator was probably placed at the top of the structure, instead of beneath the folder. Select the "Insert issues: issuetype = Epic" row and drag it to beneath the Agile Hierarchy folder (see the picture above).

5.3.4 Step 4: Insert Issues and Sub-tasks

Make sure the Agile Hierarchy folder is still selected, return to the **Automation** menu, and add the following two generators:

- **Extend | Stories under Epics...**
- **Extend | Sub-tasks...**

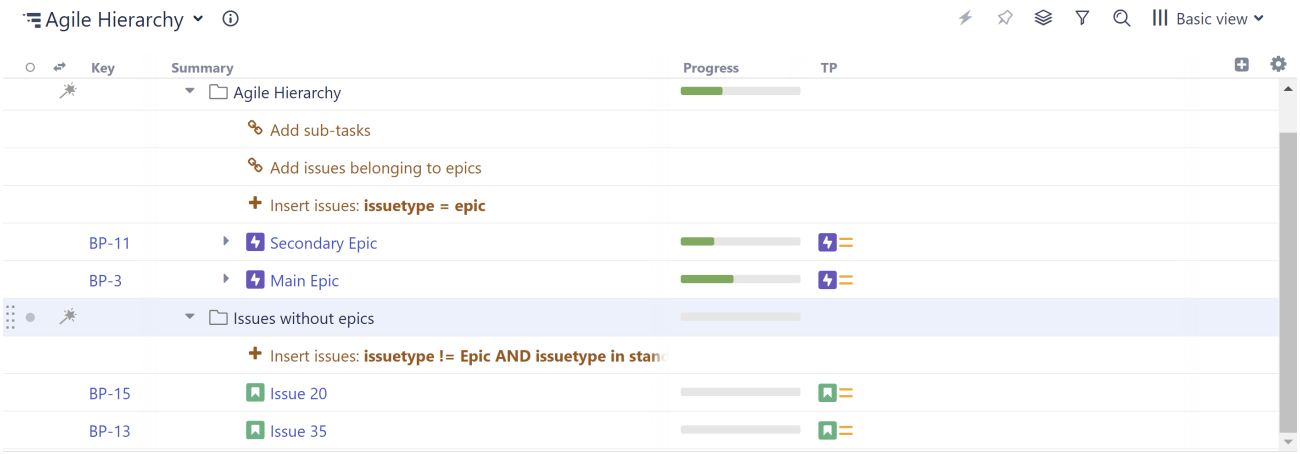


5.3.5 Step 5: Insert Issues without Epics

Next, we need to make sure we don't lose track of any issues that aren't linked to epics yet. That's what our second folder is for:

1. Select the "Issues without Epics" folder
2. Go to **Automation | Insert | JQL Query**
3. Enter the following JQL query: `issuetype != Epic AND issuetype in standardIssueTypes() AND "Epic Link" is empty`
    - If you narrowed your query to specific projects, etc. for your epics, do the same here
4. Click **Create**

Any issues that are not assigned to epics should now appear beneath the second folder.



If no issues appear under the Issues without Epics folder, make sure the Insert Generator is located in the right location - if not, drag the generator to beneath the Issues without Epics folder (see the picture above).

5.3.6 Step 6: Assign Issues to Epics / Reassign Issues

Now that you have all your issues organized and in one place, you can assign issues from the second folder to epics, or even reassign issues that already have epics.

5.3.6.1 Assigning Issues to Epics

To assign issues in the second folder to an epic:

- Drag the issues to the epic you want to assigned them to.
- While dragging, hold down the **ctrl** key - this copies the issue to the new location in the structure. Once the issue is copied to the new location, it should disappear from beneath the second folder (because now it's assigned to an epic).

Don't worry: copying an item in Structure does not make a copy in Jira.

5.3.6.2 Reassigning Issues to New Epics

To reassign issues from one epic to another, simply drag them from beneath their current epic to the new epic.

5.4 Release Management with Structure

With Structure, you can track all the issues for upcoming releases – even if they're from different projects – and visualize all the data you need to track on a single screen.

5.4.1 Step 1: Build a Release Management Structure

To create a structure for release management:

1. Go to the Jira menu and select **Structure | Create Structure**
2. Add the issues you want to track – you can add them manually or automatically using **Automation | Insert**
3. Group issues by fix version: **Automation | Group | type "Fix Version/s"** *(Tip: If you use the same names in different projects, use **"Version name..."** instead.)*
4. Open the Summary settings and select **Show Sprint and Version attributes**

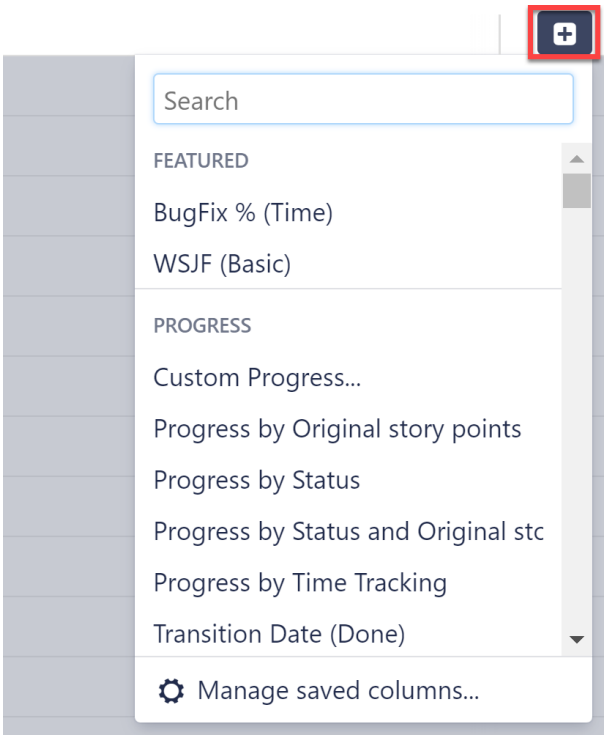
5.4.2 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate
- Progress[160] - track status based on issue progress, percent completion, or resolution
- Tempo Work Logged[161] - track how long your team is spending on each issue
- Formulas[162] - create your own formulas to compare fields or create a visual report
- Time in Status[163] - see how much time issues spend in a particular status

160 https://wiki.almworks.com/display/structure/Progress+Column
161 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
162 https://almworks.com/structure/help/column/formula
163 https://wiki.almworks.com/display/structure/Time+in+Status+Column

- Last Comment[164] - view the latest comment for each issue



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

5.5 Reporting with Structure

With Structure, you can easily create a visual overview of your projects – and visualize all the data you need to track on a single screen.

5.5.1 Step 1: Build Your Structure

To create a new structure:

- Go to the Jira menu and select **Structure | Create Structure**
- Add the issues you want to track – you can add them manually or automatically using **Automation | Insert**
- Add related issues - you can do this using **Automation | Group** and **Automation | Extend**

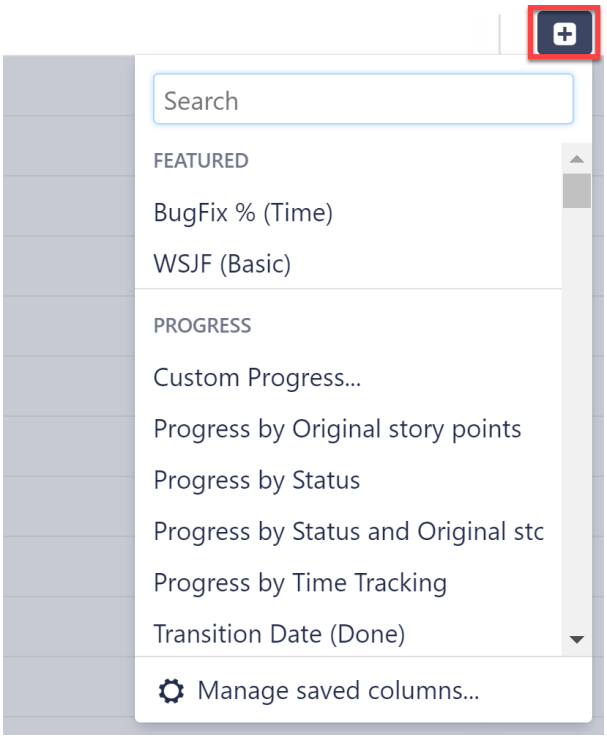
5.5.2 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate

164 https://almworks.com/structure/help/column/last-comment

- Progress[165] - track status based on issue progress, percent completion, or resolution
- Tempo Work Logged[166] - track how long your team is spending on each issue
- Formulas[167] - create your own formulas to compare fields or create a visual report
- Time in Status[168] - see how much time issues spend in a particular status
- Last Comment[169] - view the latest comment for each issue



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

5.6 Resource Allocation with Structure

When team members are working on multiple projects, it can be difficult to track how much work everyone is doing. Using Structure, you can track allocation across projects and reassign work on the fly.

5.6.1 Step 1: Build a Resource Allocation Structure

Insert issues from every project/board the team members work on, and then group them by Progress and Assignee.

To create a new structure:

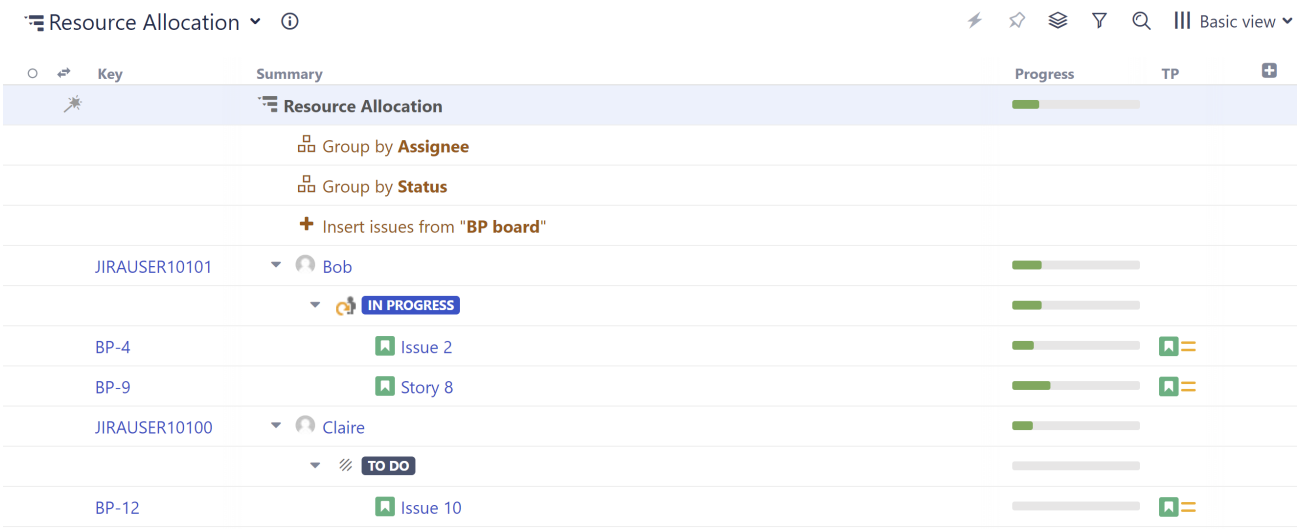
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add all the issues: **Automation | Insert**

---

165 https://wiki.almworks.com/display/structure/Progress+Column
166 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
167 https://almworks.com/structure/help/column/formula
168 https://wiki.almworks.com/display/structure/Time+in+Status+Column
169 https://almworks.com/structure/help/column/last-comment

- Select **Agile Board** to add issues from select boards
- Select **JQL Query** to add issues from projects, using JQL

3. Group issues by Status: **Automation | Group | type "Status"**
4. Group issues by Assignee: **Automation | Group | type "Assignee"**



If the status is appearing above the assignee, drag the "Group by Status" row under the "Group by Assignee" row.

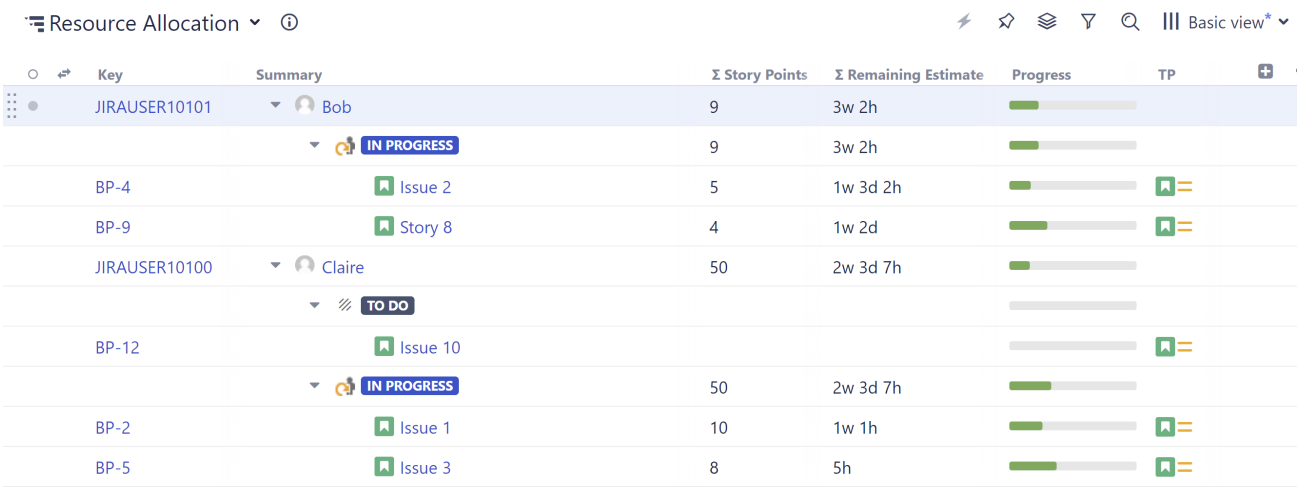
This will create a grouping for every team member with at least one issue in the current structure assigned to them. If someone doesn't have any issues assigned to them, they won't appear here. To fix this, simply assign an unassigned issue to that team member, and a new grouping will appear.

5.6.2 Step 2: Add Columns to Compare Workloads

Structure columns allow you to visualize multiple data points in a single view. We recommend one or more of these:

- Status
- ∑ Story Points (if you're using them)
- ∑ Original Estimate
- ∑ Remaining Estimate
- A custom formula(see page 463) to calculate the remaining effort

Using a Totals columns, the values for individual issues are aggregated up to their assignee, so you can evaluate workloads at a glance.



5.6.3 Step 3: Drag Issues to Reassign

To assign/reassign tasks, simply drag issues from one assignee grouping to another.

5.6.4 Optional Enhancements:

- To allocate resources at the group level, group issues by a custom Team field instead of Assignee.
- Add a WSJF column(see page 257) to prioritize issues, and select Sum over sub-items to balance priorities across resources.
- Manage allocation across sprints by adding a Group by Sprint generator. (Haven't assigned sprints yet? Check out Sprint Planning with Structure(see page 657)!)
- Consider using Structure.Gantt[170] - it can identify and resolve overallocation automatically!

5.7 SAFe (Scaled Agile Framework) with Structure

Everyone uses the Scaled Agile Framework (SAFe) a bit differently, but Structure is highly customizable and easy to tailor to your specific SAFe needs.

5.7.1 Step 1: Build a SAFe Structure

To create a structure for SAFe:

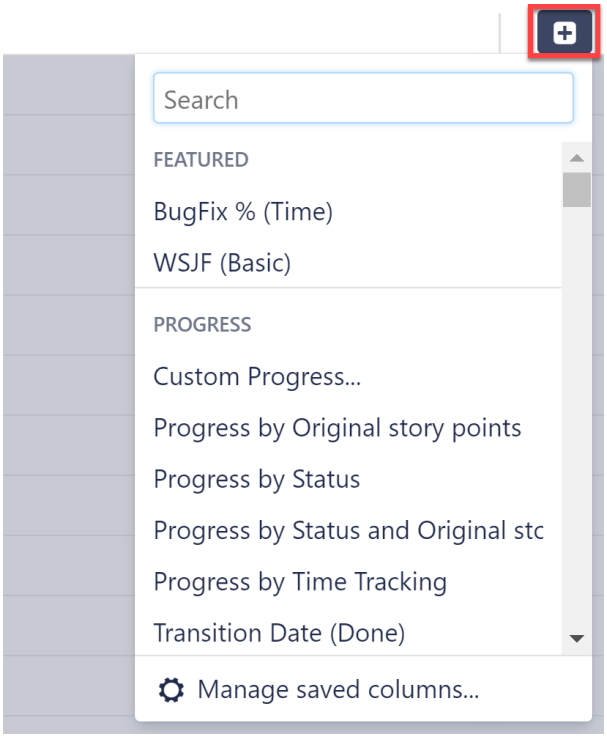
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add initiatives: **Automation | Insert | JQL Query |** enter the appropriate JQL, such as `"Project = 'project name' AND issuetype = initiative"`
3. Add epics below initiatives: **Automation | Extend | Linked Issues |** choose the type and direction used to assign Epics to Initiatives, such as "Implements" or "parent is Implemented by sub-issue"
4. Add issues belonging to epics: **Automation | Extend | Stories under Epics**
5. Add sub-tasks: **Automation | Extend | Sub-tasks**
6. Sort by rank: **Automation | Sort | type "Rank"**

170 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale

5.7.2 Step 2: Add Data

Next, add the data you need to track by clicking the **+** button at the top-right corner of the structure. You can add as many columns as you need, including:

- Jira fields, such as status, assignee, sprint, or remaining estimate
- Progress[171] - track status based on issue progress, percent completion, or resolution
- Tempo Work Logged[172] - track how long your team is spending on each issue
- Formulas[173] - create your own formulas to compare fields or create a visual report
- Time in Status[174] - see how much time issues spend in a particular status
- Last Comment[175] - view the latest comment for each issue



Once you've added your columns, you can aggregate the data up the structure. To do so, open the column you want to aggregate and select **Sum over sub-items**.

Take SAFe management to the next level

By adding Structure.Pages[176], you can manage your Strategy documentation and linked issues all in one place.

Learn More[177]

171 https://wiki.almworks.com/display/structure/Progress+Column
172 https://wiki.almworks.com/display/structure/Tempo+Work+Logged
173 https://almworks.com/structure/help/column/formula
174 https://wiki.almworks.com/display/structure/Time+in+Status+Column
175 https://almworks.com/structure/help/column/last-comment
176 https://alm.works/pages
177 https://almworks.com/structure/help/how-to/safe

5.7.3 Advanced SAFe (Scaled Agile Framework) with Structure

The following guide will walk you through an advanced SAFe implementation in Structure.

5.7.3.1 Requirements

To do this, you will need:

- Jira Software and Confluence
- Structure
- Structure.Pages
- Structure.Testy

Ready-to-use demo space: Live Demo[178]

5.7.3.2 Before You Begin

We assume that your Jira is prepared for scaling. The most common configuration is to create separate projects for each level:

- Kanban project for Portfolio level, with the following Epic issue type and statuses: Funnel, Reviewing, Analyzing, Portfolio Backlog, Implementing and Done
- Kanban project for Solution level, with the following Issue types: Capability and Enabler
- Kanban project for Program level, with the following issue types: Feature and Enabler

> ⚠ Features are named "Epics" in Jira, so you can rename them in Administration. Projects for each Team (Scrum or kanban, Scrum is preferable) and Issue types (User Story, Bug, Improvement, Enabler) can be left as is.

5.7.3.3 Step-by-step guide

1. **Define your company Strategy in Confluence**
   Create a Space in Confluence for high-level documents and describe the Strategy of your company.

2. **Define Themes**
   Define Strategy more precisely by dividing it into several themes, each a sub-page of Strategy.

3. **Link Themes to Epics**
   Define Epics, or high-level tasks for implementing your theme. Link Jira issues with their corresponding themes in Confluence.

4. **Add Capabilities**
   Since you have a separate project for Capabilities, divide your Epics into more precise Capability issues and put them into a Capability Board. Don't forget to link each Capability with a corresponding Epic! You can use standard Jira issue link types, but it will be more convenient to create you own[179] "implements"/ "is implemented by" link type.

178 http://alm.works/safe-demo
179 https://confluence.atlassian.com/adminjiraserver071/configuring-issue-linking-802593137.html

5. **Add Features**
   Add details to any large Capabilities and divide them into Features. Don't forget to link each Feature to its corresponding Capability.

6. **Stories**
   Each Feature can be divided into more detailed Stories. Don't forget to link Features and Stories with an epic link.

7. **Sub-tasks**
   If you need more detailed tasks, divide Stories into Sub-tasks.

8. **Put it all together with Structure**
   Create an empty Structure and visualize everything in one place: http://recordit.co/lFYAzeBezV
   - Find your Strategy page and add it to the Structure: http://recordit.co/0vt6mNYsqR
   - Click the Automation button to enable automation. Then click the **+** icon next to the **Automation** button, and use the Extend generator to add the following rules:
     - Add all child pages (extend | child pages): http://recordit.co/5l5fqPM4c3
     - Add issues linked to pages
     - Add Capabilities linked to Epics
     - Add Features linked to Capabilities
     - Add Stories linked to Features
     - Add add Sub Tasks linked to Stories

9. **Take it to the next level!**
   Now that your structure is created, you can modify it to fit your needs:
   - You can Group issues on a specific level. For example, group by Sprint on the Team level to track progresses.
   - You can prioritize issues by sorting by WSJF.
   - You can specify your Objective during PI planning in Notes.

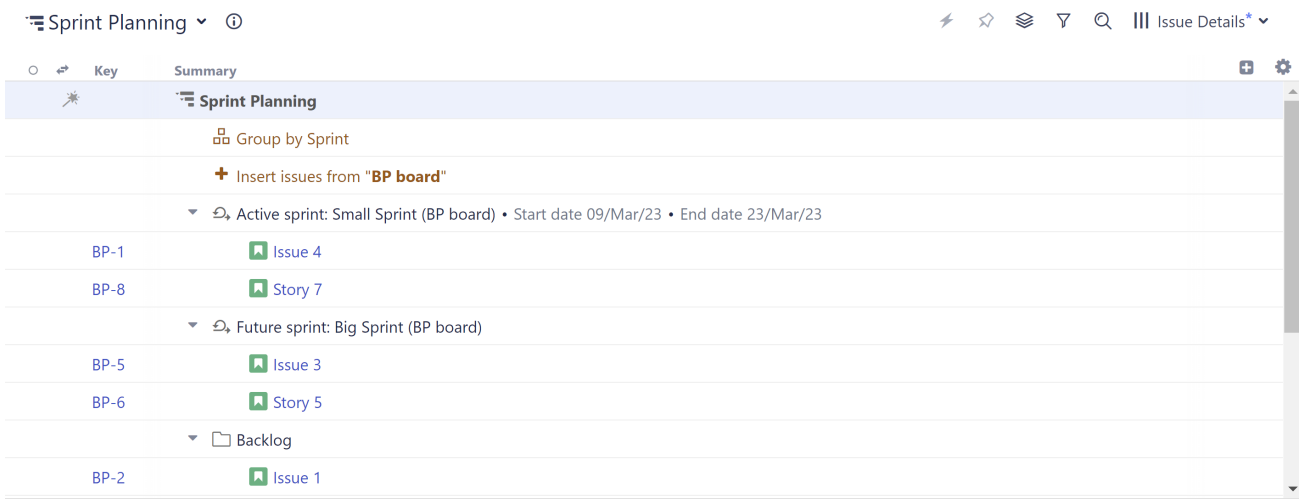
5.8 Sprint Planning with Structure

With Structure, you can track all the issues for upcoming sprints – even if they're from different projects – and visualize all the data you need to track on a single screen. You can even aggregate that data for each sprint.

5.8.1 Step 1: Build a Sprint Planning Structure

To create a new structure:

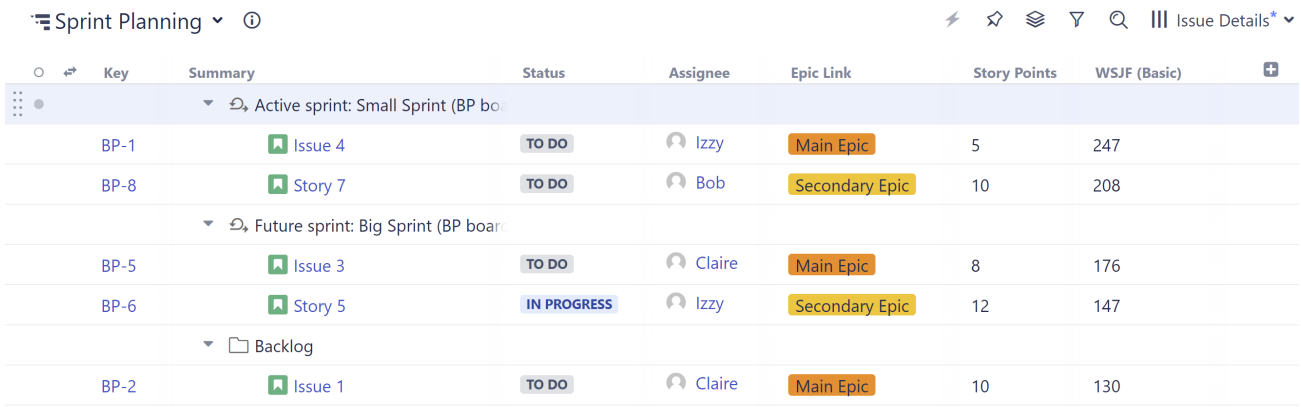
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add all the issues from a Board: **Automation | Insert | Agile Board**
3. Group by sprints: **Automation | Group | type "Sprint..."**
4. Open the Summary settings and select **"Show Sprint and Version attributes"**:



5.8.2 Step 2: Add Data

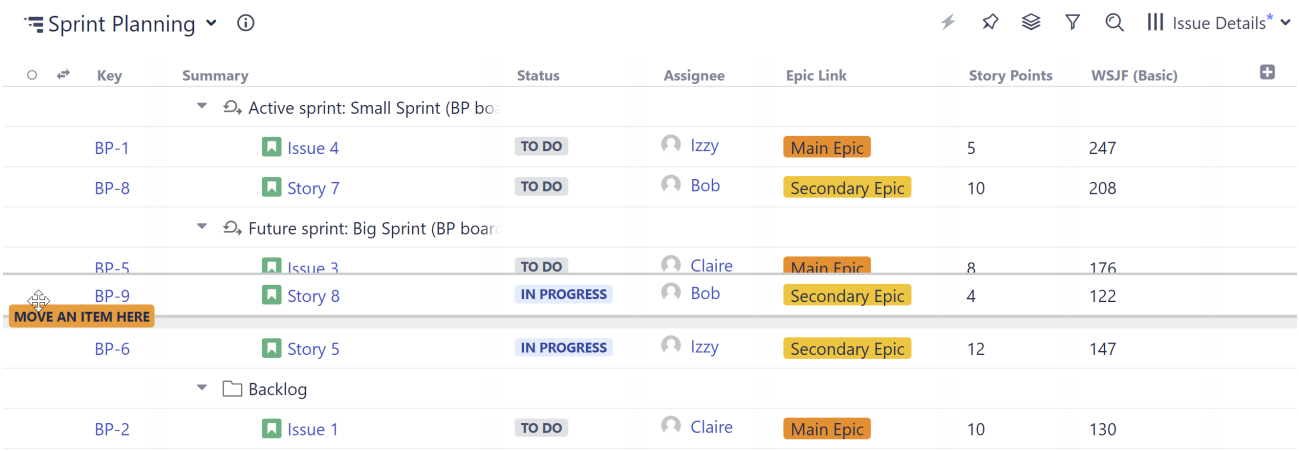
Next, add the data you'll use to compare issue priority:

1. Click the **+** button at the top-right corner of the structure
2. Select the fields or attributes you want to compare - we recommend:
    - Status
    - Assignee
    - Epic Link
    - Story Points (if you're using them)
    - WSJF (or any other metric your team uses to prioritize issues or assess business value)
3. Click the header of any column to sort your issues by that value.



5.8.3 Step 3: Assign Issues

To assign issues to a sprint, drag it from the backlog to the appropriate sprint. You can also reassign issues by dragging them from one sprint to another.



5.8.4 Optional Enhancements

- Aggregate values for each sprint: when adding a column (such as Story Points), select **Sum over sub-items**
- Group by assignee under each sprint to see how the work is divided across your team: **Automation | Group | Assignee**
- Hide closed sprints: **Automation | Filter | Hide Closed Sprints**
- Use Structure.Gantt[180] to view any blocking dependencies between issues (to help prioritize them)

5.9 Track Overdue Tasks with Structure

Use Structure to identify:

1. Overdue tasks
2. Number of overdue tasks per epic, initiative, or other larger grouping
3. Ratio of overdue tasks to total tasks for each epic, initiative, or other large grouping

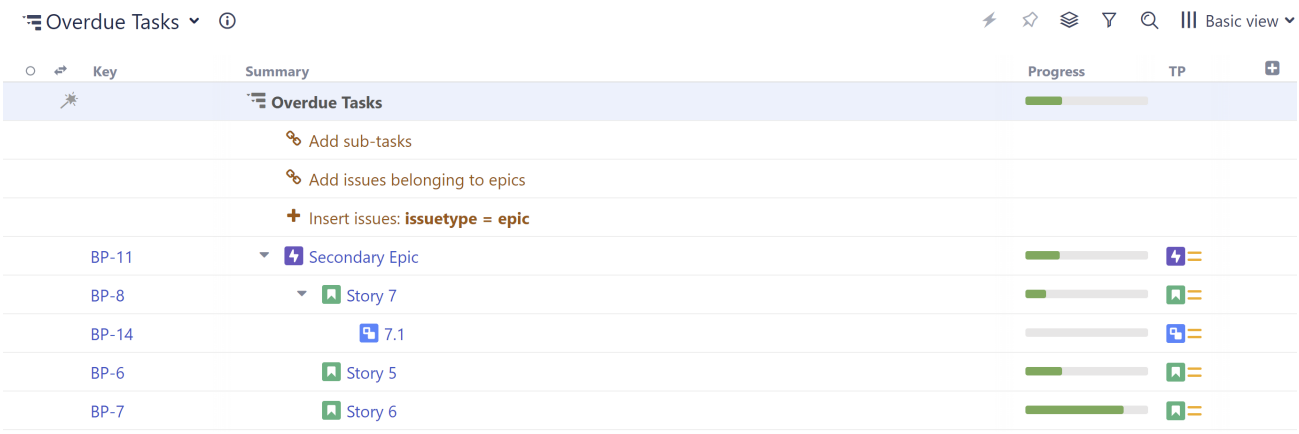
5.9.1 Step 1: Build Your Structure

In this guide, we're going to show you how to track overdue tasks at the epic level. To track overdue tasks for initiatives or any other large groupings, simply reconfigure the structure hierarchy - this can be done using different Insert(see page 142), Extend(see page 148), and Group (see page 169)generators.

To create a new structure:

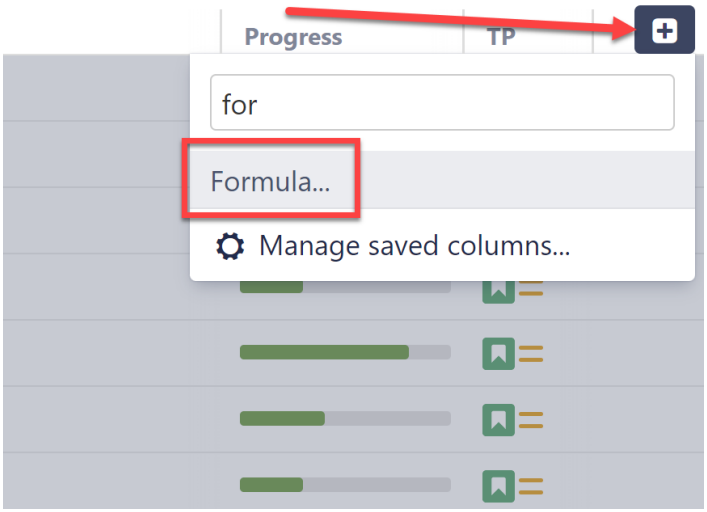
1. Go to the Jira menu and select **Structure | Create Structure**
2. Add epics: **Automation | Insert | JQL Query** and enter the following JQL query: `issuetype = Epic`
    - To limit the epics to specific projects or other variables, add additional specifications (Example: `AND project = "My Epics"`)
3. Add stories: **Automation | Extend | Stories under Epics...**
4. Add sub-tasks (optional): **Automation | Extend | Sub-tasks...**

180 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale



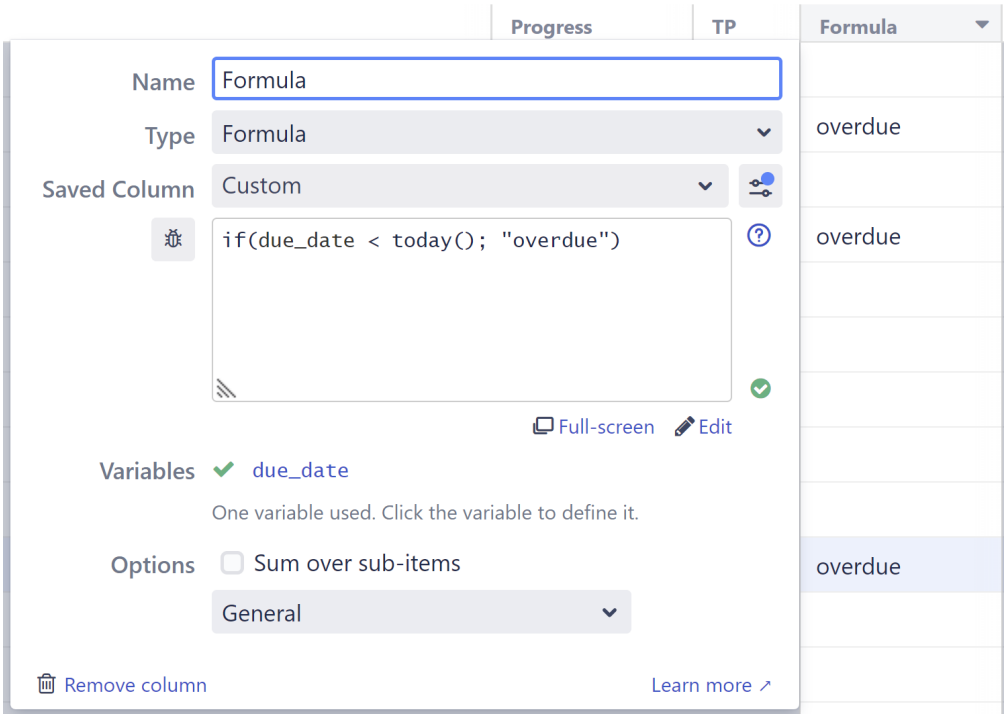
5.9.2 Step 2: Mark Overdue Tasks

Once you have your structure in place, the next step is to identify all the overdue items. To do this, we need to add a formula column(see page 659). Click the **+** button to the right of the column headers, and select **Formula.**



Give the column an appropriate name and enter the following into the Formula section: `IF(due_date`[181] `< today(); "overdue")`

181 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6



If you used "due_date" in your formula, the variable will automatically be mapped to the Due Date attribute. If you used another name for that variable (for example, “due”), you will need to define the variable by pointing it to the appropriate Due Date attribute.

5.9.2.1 Variations

This is just one way to mark overdue items. You could also:

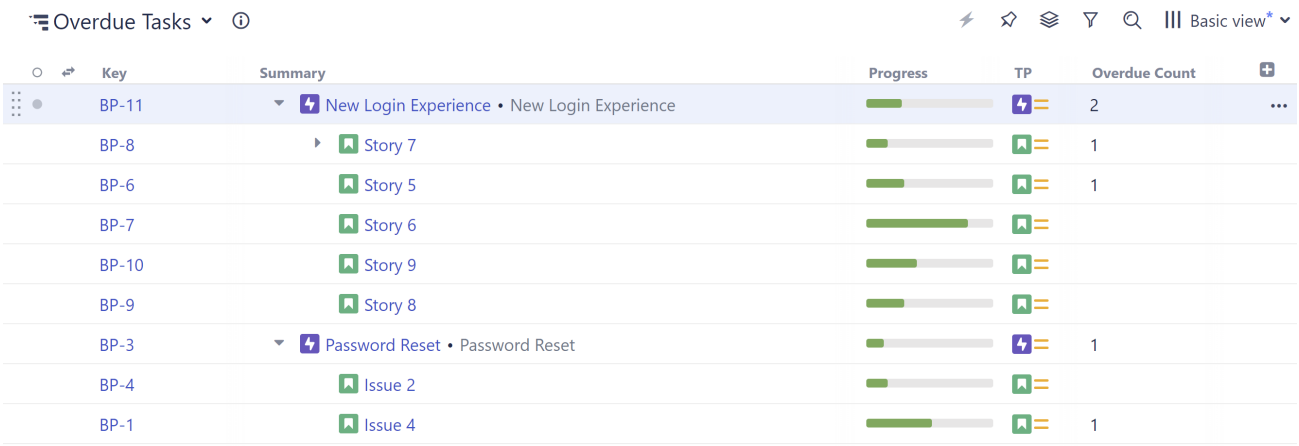
- Create a more visual warning that highlights not only overdue items, but also items coming due. You can read how to do this in Wiki Markup in Formula Columns(see page 251).
- Create a formula that relies on another due date. For example, you may want to flag issues that aren't completed by the due date of their parent epic. In this case, change the formula to: `IF(type[182] = "Story" and status[183] != "Done" and parent{due_date[184]} < today(); "overdue")`
- Aggregate overdue items for epics (we'll cover this next!)

5.9.3 Step 3: Aggregate Overdue Items for Epics, Initiatives, etc.

In the previous example we were simply returning the string “overdue”, but if we give that "overdue" flag a numeric value ("1"), we can use the Sum over sub-items option to aggregate those values up the hierarchy. To accomplish this:

182 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
183 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
184 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6

1. In place of the formula we created in step 1, enter the following: `IF(type`[185] `= "Story" and status`[186] `!= "Done" and due_date`[187] `< today(); 1)`
2. Check the Sum over sub-items box



In this example, we limited our scope to only stories that are not yet "Done" - depending on your situation, you may want to adjust this to include other issue types, add additional qualifications, etc.

5.9.4 Step 4: Calculate the Percentage of Stories That Are Overdue

So far, we've just identified and counted our overdue stories. But this doesn't paint a complete picture. And epic with 1 overdue item out of 257 is very different from 1 out of 5!

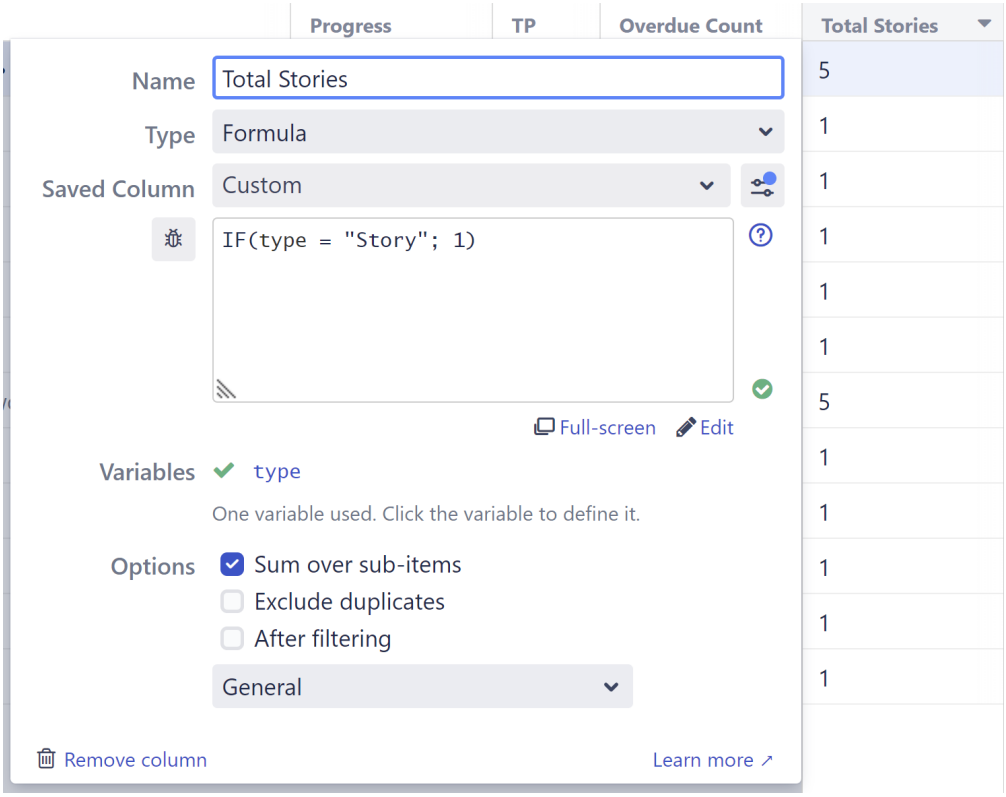
To get a better understanding of how these overdue items relate to the big picture, next we're going to:

- Calculate total stories for each higher level in our hierarchy
- Calculate the percentage of stories that are overdue for each higher level

First, we need to create another Formula column, which we'll use to calculate our total stories. For this column:

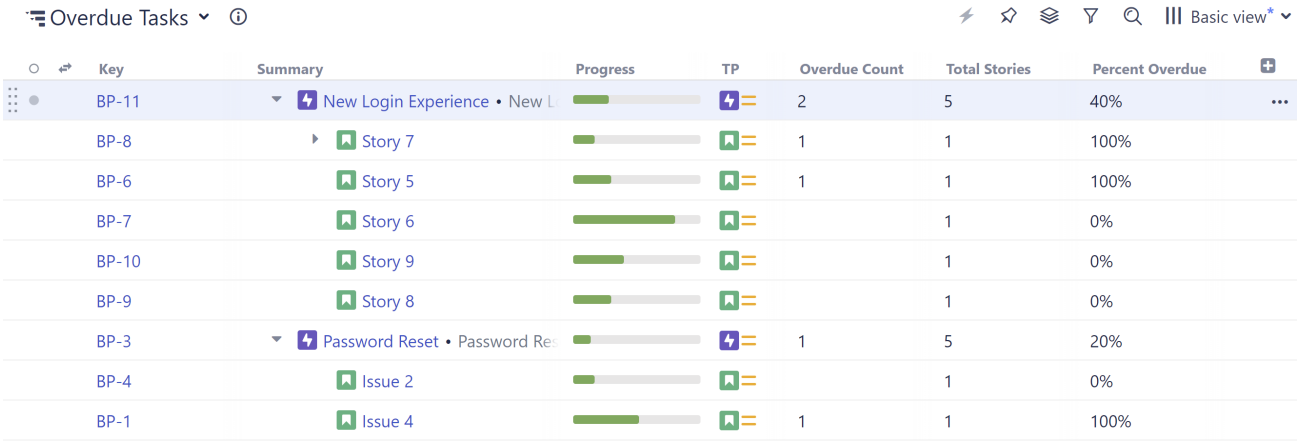
- Enter the following formula: `IF(type = "Story"; 1)`
- Check the Sum over sub-item box

185 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
186 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
187 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6



Next, we'll create a third formula column to compare our values from the first two. For this column:

1. Enter the formula: `IF(total_stories`[188]`; overdue`[189]` / total_stories`[190]`)`
2. Map your "total_stories" and "overdue" variables to the appropriate columns we created above



> For this formula, we wrapped the expression in an if statement so we wouldn't get a DIV/0 error on lines without a value for total stories.

188 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
189 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
190 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6

At this point, you can keep all three columns in your structure (we recommend saving them as a new view(see page 511)), or get rid of one or both of your original columns - our Percent column will keep their original calculations, even if those columns are deleted or changed. (This also means if you make changes to the formulas used in the Overdue or Total Stories column, you'll need to re-select those columns in the Variables section of your Percent formula to apply those changes there.)

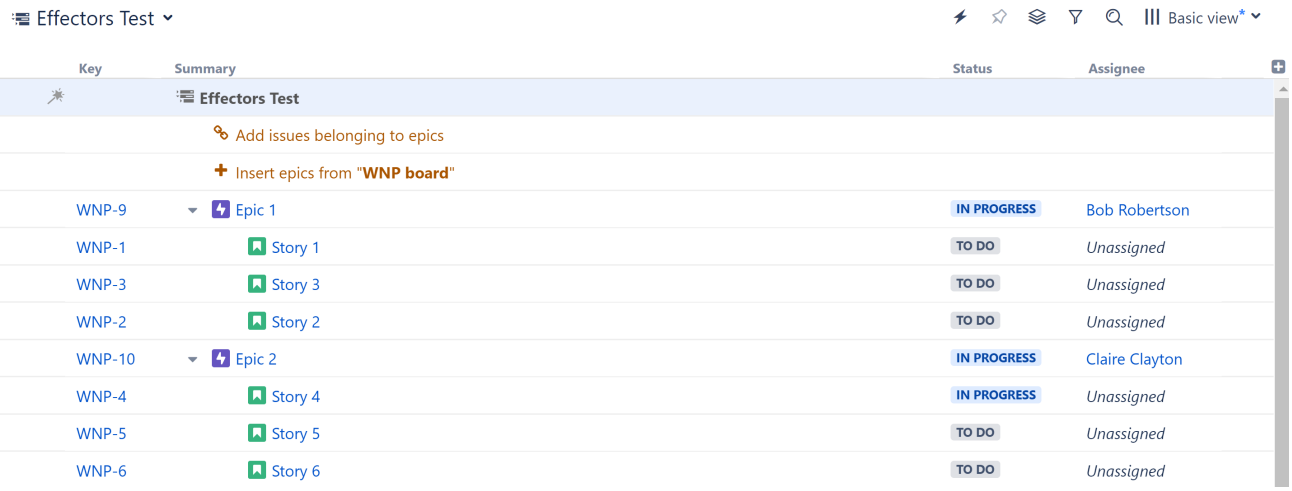
5.10 Update Assignees to Match Parent Issues

Use Structure to automatically reassign issues based on a parent item - for example, update the assignee for each story to match the assignee of its parent epic.

5.10.1 Step 1: Create the Hierarchy

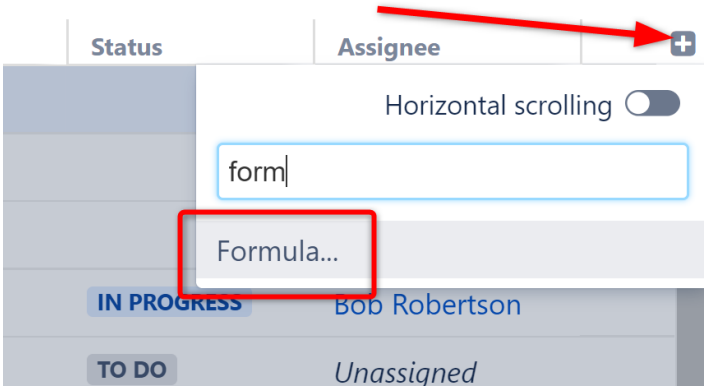
Use Automation(see page 140) to organize issues into a hierarchy so that the issues you want to update are placed directly beneath the issues you want to match. For this example, we're going to place our stories beneath our epics:

1. Add epics: **Automation | Insert | JQL Query**
   - Enter the following JQL query: `issuetype = Epic`
   - To limit the epics to specific projects or other variables, add additional specifications (Example: `AND project = "My Epics"`)
2. Add stories: **Automation | Extend | Stories under Epics...**

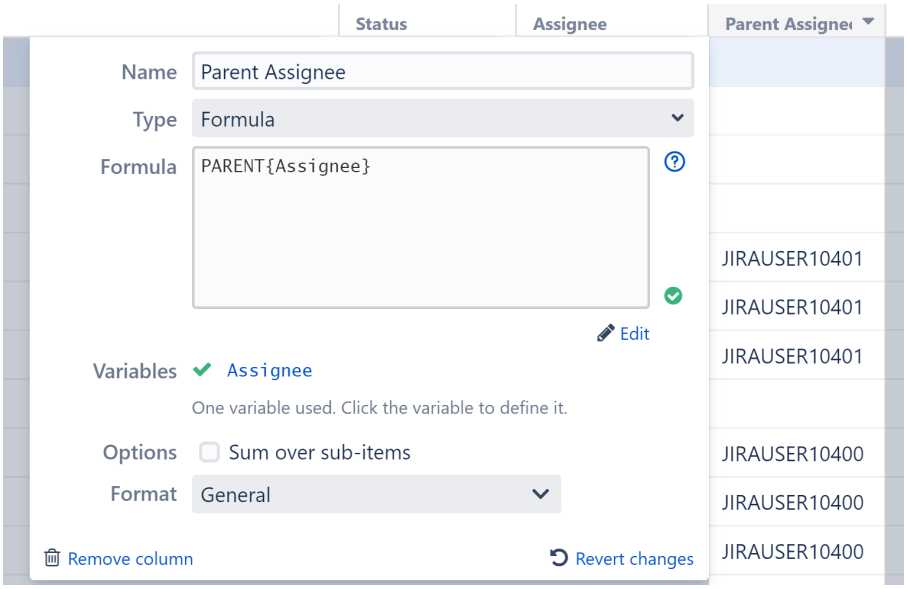


5.10.2 Step 2: Add a Formula Column to Capture the Parent's Assignee

Click the **+** icon to the right of the column headers, and select **Formula...**



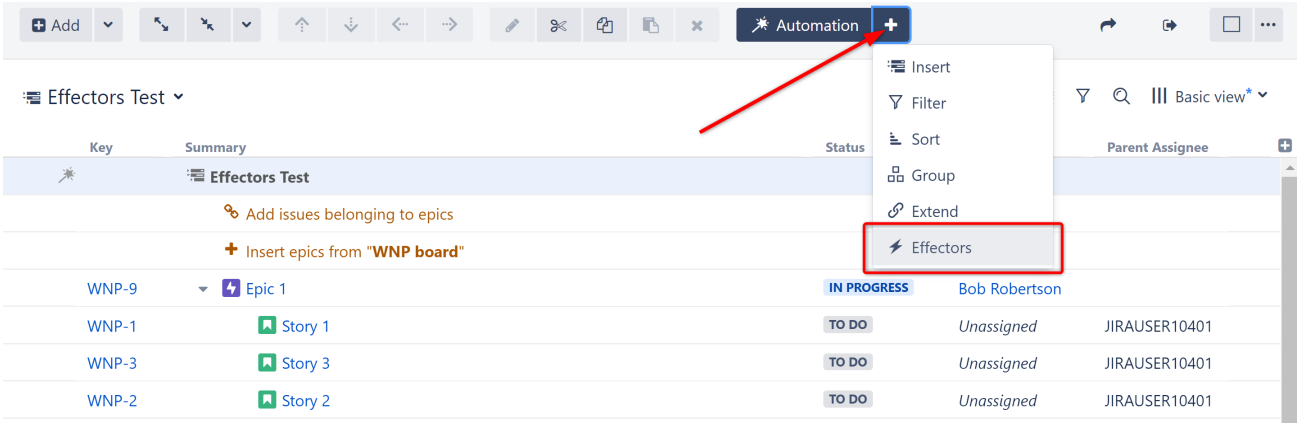
Give the column an appropriate name, and enter the following formula: `PARENT{Assignee}`



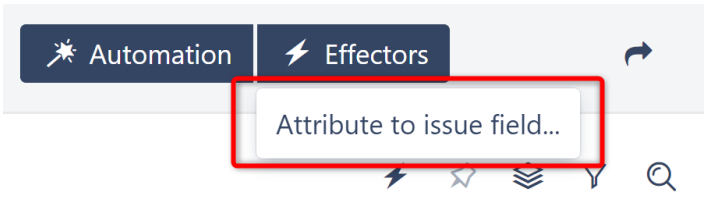
5.10.3 Step 3: Add an Effector

Now we need to take the information from our formula (which is only available in Structure) and write it to the Assignee field (so it is available anywhere in Jira).

Open the Automation menu and select **Effectors**.

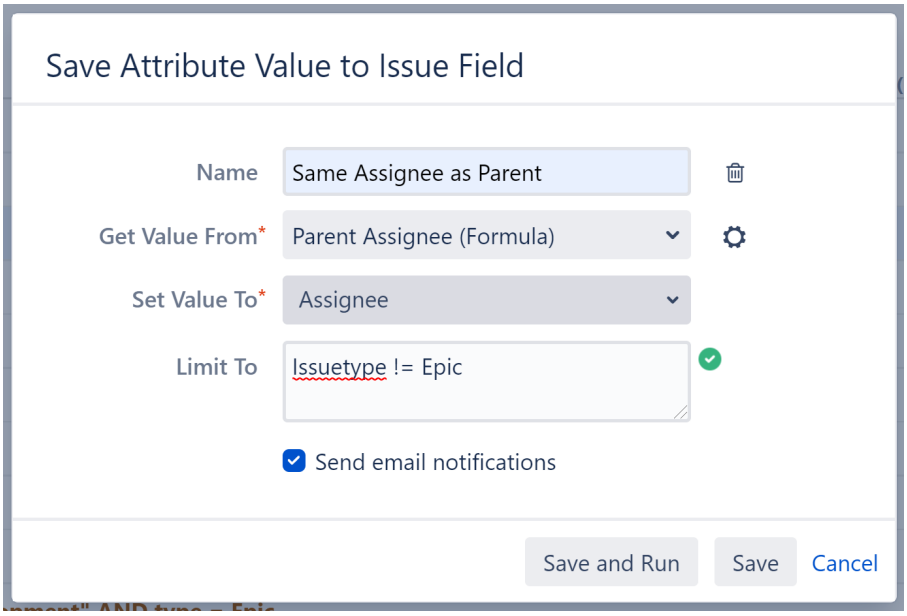


Choose **Attribute to issue field...**



On the Effector settings screen:

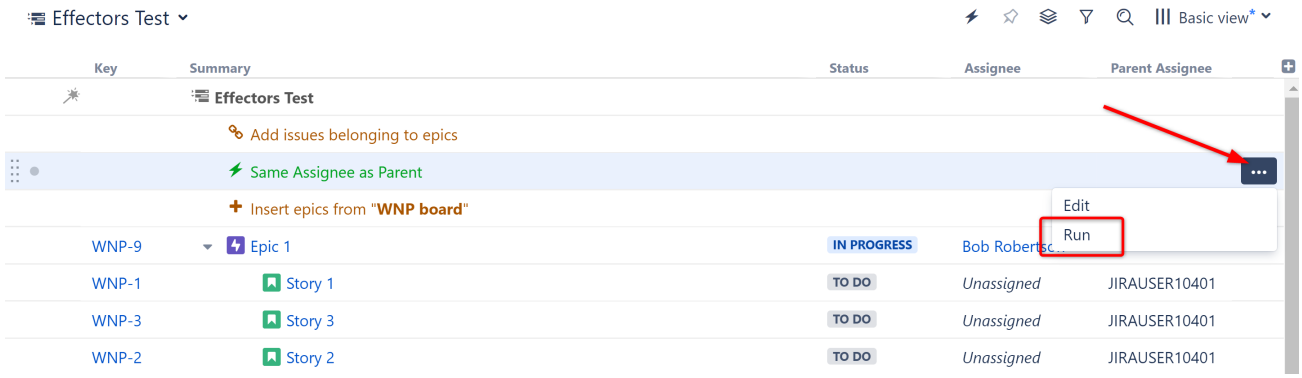
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Get Value From: Select the formula column we created in Step 1
- Set Value To: Assignee
- Limit To: Since we're getting our Assignee value from Epics, we want to make sure their Assignee value isn't changed - so we've added the following JQL Filter: `Issuetype != Epic`
- Select whether email notifications should be sent when the Effector writes values to Jira



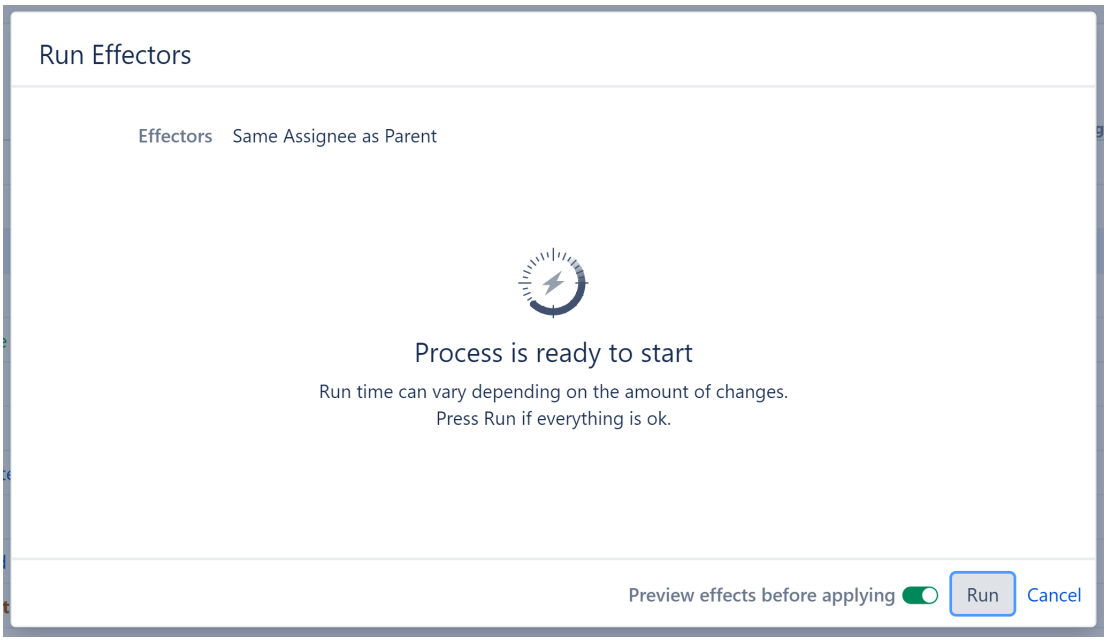
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

5.10.4 Step 4: Run the Effector

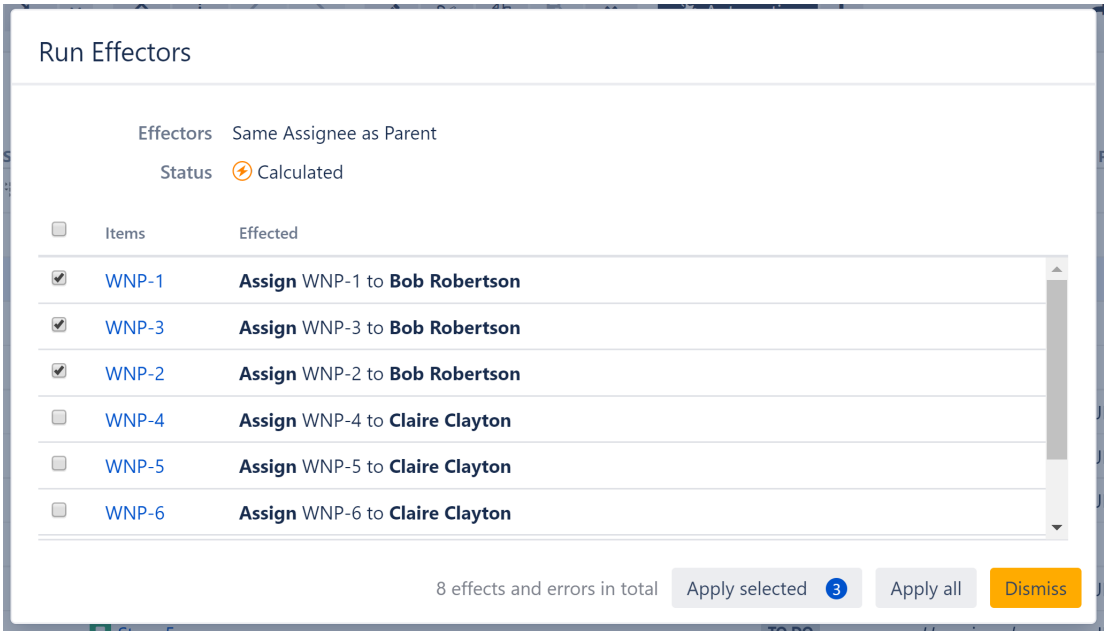
You can run an Effector directly from its settings screen (see above), or you can locate the Effector at the top of the structure, click the Action button (the three dots to the right of its row) and select **Run**.



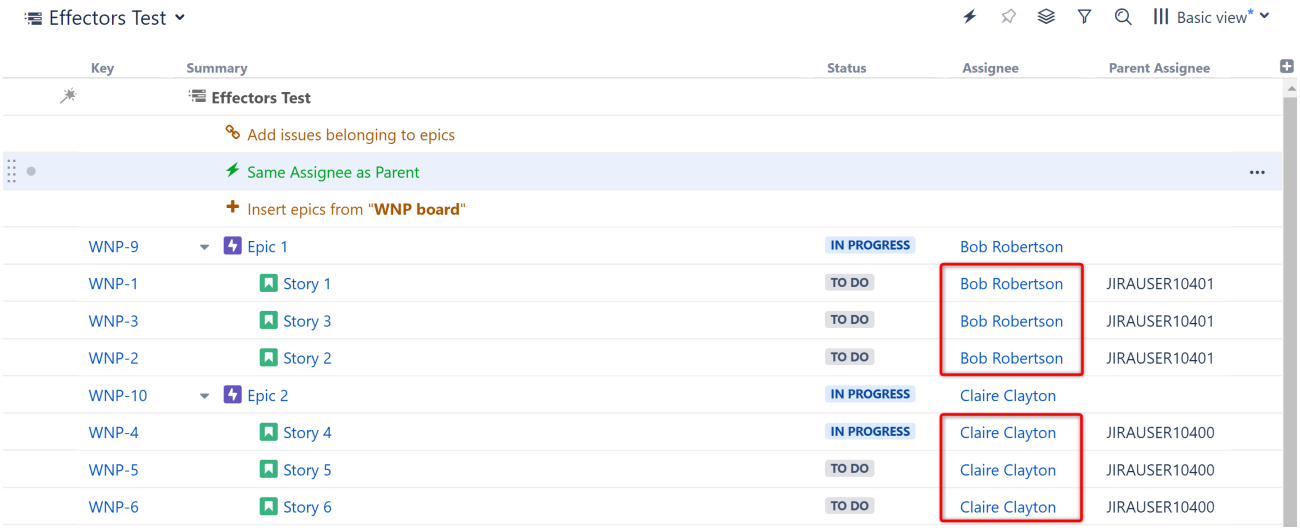
You have the option to **Preview effects before applying**. This allows you to view and approve every change that will be made by the Effector. Effectors update live Jira data, so we highly recommend using the preview option. Click **Run** to begin.



Once the preview is finished, you will see a list of changes that will be made by the Effector. You can select which changes you want made, or click **Apply All** to apply all changes.



Once you've applied the selected changes, the Assignee field for each child issue should be updated to match its parent's Assignee:



5.11 Update Link Types for Multiple Issues

When you're working across multiple projects and teams, chances are there are some inconsistencies between the link types being used.

The following guide will show you how to quickly update all your links to a single, consistent link type using Structure.

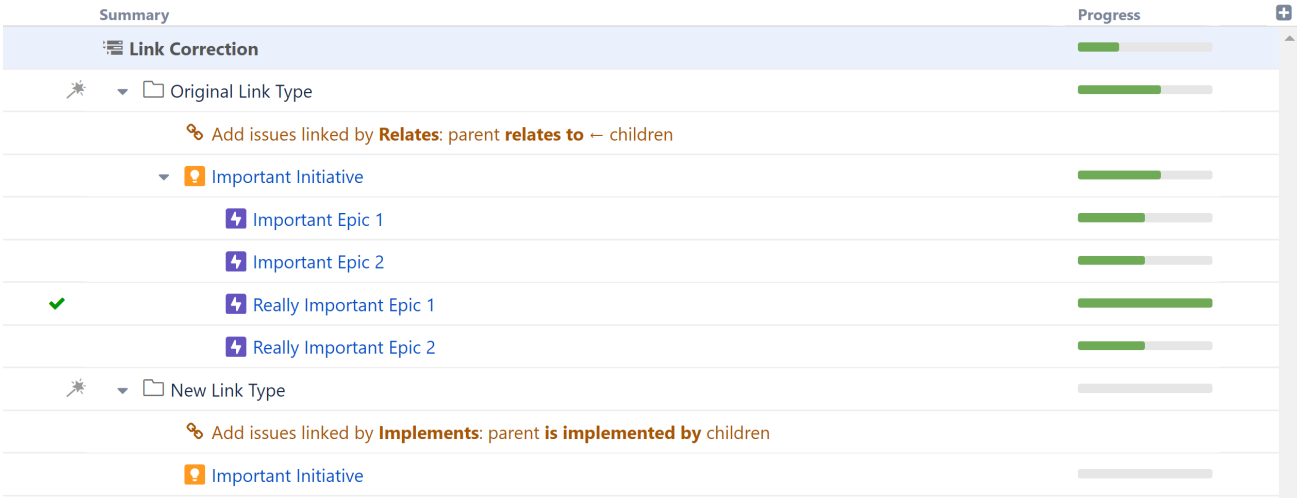
> The examples below demonstrate how to update link types for an initiative, but the same process can be used for other link types as well.

5.11.1 Step 1: Build a Simple Structure

1. Create a new, blank structure and add 2 folders: "Original Link Type" and "New Link Type".
2. Manually add your parent initiative to both folders. *Note: this should be the same initiative for both.*

5.11.2 Step 2: Add Epics

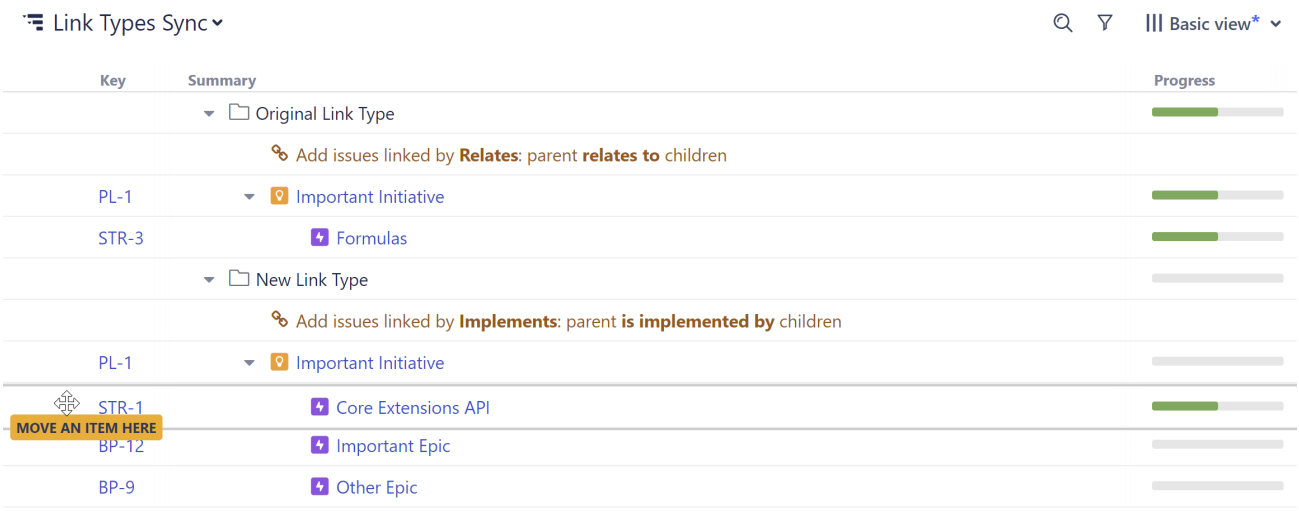
1. Highlight the Original Link Type folder and add a Linked Issue generator: **Automation | Extend | Linked Issues |** select the current (inconsistent) link type
2. Highlight the New Link Type folder and add a Linked Issue generator: **Automation | Extend | Linked Issues |** select the desired link type



⚠ Make sure the **Allow changes via Structure** option is checked for both generators.

5.11.3 Step 3: Update Link Types

Select all the issues under the generator added to the first folder, and **drag them to the second folder**, as children of the same initiative.



Structure will automatically replace all the old links with the new link type.

> Need to keep the original links too? When dragging the items, hold down the "ctrl" key (Option key on Mac) to copy the items to the new iniative.

5.11.4 Additional Use Cases

You can also use this same method to:

- Update other link types - just edit your Extend | Link generators and the original item added to each folder
- Remove links - include an empty folder (no Extend generator) and drag your linked issues to the empty folder - the original link will be removed without adding a new link

6 Frequently Asked Questions

6.1 Cannot Create an Issue With +Next Issue (+Sub-Issue) Because of the Required Fields

6.1.1 Question

I have a number of fields required for new issues. When I try to use Structure's **+Next Issue** or **+Sub-Issue** button, the creation of the issue fails, because the values of the required fields were not provided.

6.1.2 Answer

There are two ways to create new issues when additional fields are required:

6.1.2.1 1. Switch to the Create Issue Dialog

At the bottom of the Add Issue panel, click the Switch to Dialog link to access additional fields. See Creating New Issues: Create Issues Using the "Create Issue" Dialog(see page 130) for more information.

6.1.2.2 2. Add the Required Field to the View

You can enter additional fields when creating a new issue, as long as the field is part of the current view. See Creating New Issues: Editing Other Fields During Creation(see page 129) for more information.

If the initial creation of an issue has failed, you don't have to lose the entered data. Just add the required fields and double-click on the value you need to edit, or click the **Edit** button in the toolbar. You can change the values of the new issue and try to create it again.

6.2 Add-on Manager Says Structure Is Unlicensed

6.2.1 Question

I have a valid license installed. Why do I see Structure as **Unlicensed** or having **Action Required** in the Add-on Manager?

6.2.2 Answer

That may be so because Add-on Manager is not aware of ALM Works licenses. To verify the true status of your Structure license, check the **Administration | Structure | License Details** page. If it indicates the license is OK, you can safely ignore the status of the Structure license in Add-on Manager.

Structure supports two kinds of licenses — purchased via Atlassian and issued by ALM Works. For details, please see Setting Up Structure License(see page 950).

6.3 No Check Mark Displayed for a Resolved Issue

6.3.1 Question

Why do I see a resolved issue in Structure, but there's no green check mark, which usually indicates that an issue is resolved?

This article answer these questions as well:

- Why do I see a check mark on a unresolved issue?
- Why does an open issue that still in the work have 100% progress indication?
- When I turn on "Unresolved" filter button, why do I see some of the resolved issues anyway?

6.3.2 Answer

The JIRA's notion of a "Resolved Issue" (or "Completed Issue") can be quite confusing. The source of confusion is that an issue is considered to be resolved based on its **Resolution** field, not based on its Status:

- **Unresolved** means that the Resolution field is empty, regardless of issue Status.
- **Resolved** means that the Resolution field has some value, regardless of issue Status.

If an issue has a non-empty Resolution field (i.e. considered Resolved):

- The green check mark is displayed in Structure on that issue;
- The issue is filtered out by the Unresolved button;
- The progress of the issue is 100% regardless of other fields.

See also: Flags Column(see page 461), Filtering(see page 672), Progress Column(see page 472)

6.3.2.1 Problems Caused By Custom Workflows

The default workflow in JIRA contains the "Resolved" status and if you select this status, JIRA requires you to select some non-empty value for the Resolution field too. Thus, the issue gets the Resolved status and becomes truly resolved (or completed), because it has a value in the Resolution field.

The confusion may arise, if in a custom workflow / screen configuration, Resolution field is not set as required or not added to the screens, associated with transitions to the Resolved status. In this case, a user may move an issue to the Resolved status, but the issue will still be unresolved/uncompleted, because the Resolution field is still empty.

If you have such a configuration, in the Structure this problem may manifest itself when you are trying to use the Unresolved filter button (which works as a shorthand for filtering using JQL: "`Resolution is EMPTY`"). The issues with the Resolved status but with no Resolution will still be visible even if you switch the filter on.

Solution:

1. Edit your workflow: in all transitions to a status that should be considered resolved, use a screen with the Resolution field.
2. In all transitions to a status that should not be considered resolved, use "Clear Resolution" step.
3. Make Resolution field required. (It will matter only if Resolution is added to the screen configuration.)
4. Check all screens - "Edit Issue" screen and all screens not mentioned in (1) above should not contain Resolution field.

6.3.2.2 Problems Caused By Manually Added "Unresolved" Resolution Value

To make matters worse, sometimes JIRA administrators add a new resolution option, named "Unresolved". Then, for example, on the workflow's "Reopen" step configuration, instead of clearing the Resolution, they change it to this "Unresolved" value.

The problem is that the new "Unresolved" resolution is still a non-empty value, and any issue having this value in the Resolution field will be considered resolved, by JIRA and Structure and other plugins.

But on the issue page, the user will see *Resolution: Unresolved*. So it will be practically impossible to distinguish this resolved (completed) issue from the issues which are really unresolved (have empty Resolution field).

Solution:

1. Use JIRA's Bulk Change to clear resolution from all issues that have Resolution "Unresolved".
2. Remove resolution "Unresolved".

6.4 Structure Won't Start

6.4.1 Question

I try to install (enable) Structure, but it doesn't work. When I reload the Add-on Manager page, Structure is disabled. What is the problem?

6.4.2 Answer

Structure may fail to start due to the following reasons. To better understand what's going on, check the Jira logs (`catalina.out` or `jira-application.log`) and verify each of the following possible causes.

6.4.2.1 Structure database cannot be created or opened; filesystem is read-only or full

Structure stores all its data in the `structure/` sub-directory of the Jira home directory. At its first launch, it tries to create that directory and shuts down if it fails to do so. At every subsequent start, it tries to open the database contained there and also shuts down if it fails to do so. In all cases, there should be a big warning or error message in the Jira log.

Possible actions:

- Create the `structure` sub-directory manually and grant full permissions on it to the account that is used to run Jira.
- Verify that filesystem is not read-only.
- Verify that there's enough free disk space (at least 100 MB).
- Verify that Structure's database is not opened with some other tool, such as Derby console.

6.4.2.2 Some of the required system plugins are disabled

Structure relies on some system plugins. If they are disabled, you may get all kinds of weird messages from Jira when it tries to start Structure. In some cases, the error messages will be completely unrelated to the disabled plugins.

The following is an example of a message you may receive:

```
com.atlassian.plugin.PluginParseException: Unable to load the module's display
conditions: Could not load 'com.almworks.jira.structure.web.UserCanCreateStructureCon
dition' in plugin com.almworks.jira.structure
... stack trace ...
Caused by: com.atlassian.plugin.web.conditions.ConditionLoadingException: Could not
load 'com.almworks.jira.structure.web.UserCanCreateStructureCondition' in plugin
com.almworks.jira.structure
... stack trace ...
Caused by: java.lang.IllegalStateException: Cannot autowire object because the Spring
context is unavailable.  Ensure your OSGi bundle contains the 'Spring-Context'
 header.
... stack trace ...
```

Possible actions:

- Open **Administration | Plugins | Manage Plugins** and click **Show System Plugins**. Verify that all plugins are enabled. If some are disabled, enable them. Then try to enable or reinstall Structure.

If for some reason you need to keep some of the plugins disabled, and Structure won't start without them, please write to Tempo Support[191].

6.4.2.3 Incomplete download or corrupt plugin JAR file

It is possible the Add-on Manager only downloaded a partial JAR file for Structure, if there were any problems with the server or the connection. We have also had reports of Internet Explorer incorrectly downloading the manual install file, turning the JAR file into a ZIP file with invalid content.

---

191 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Possible actions:

- To verify that you have a correct JAR file, locate the plugin JAR in `plugins/installed-plugins` directory under your Jira home. Structure has the word "structure" in its file name. Verify that the JAR file MD5 hash is the same as listed on the Download Archive(see page 1165) page.

6.4.2.4 Incorrect Jira setup

Check the Jira application logs[192] for one or several lines that look like the following:

```
ERROR      [plugin.osgi.factory.OsgiPlugin] Unable to start the Spring context for
plugin com.almworks.jira.structure
```

In order for Structure to work, it requires some of the standard Atlassian plugins, such as the one that allows Structure to post to the Activity Streams[193]. We have had reports of cases where these plugins cannot start because the

`-Datlassian.org.osgi.framework.bootdelegation` variable was set in `JAVA_OPTS` in `setenv.sh` (`setenv.bat`), as recommended in this comment to the Upgrade to JIRA 4.2 Guide[194]. If you are using Jira 5.0 or later, please try to remove the variable from `JAVA_OPTS` and see whether it resolves the problem.

If none of the above help resolve the problem, please contact Tempo Support[195].

6.5 After an Issue is Moved to Another Project, It Cannot Be Found in the Structure

6.5.1 Question

An issue was added to the structure. Afterwards, the issue was moved in Jira to another project. Now the issue cannot be found in the structure, either by summary or by the new or old issue key. What happened?

6.5.2 Answer

There may be a couple of reasons for this:

192 https://confluence.atlassian.com/pages/viewpage.action?pageId=16121981
193 https://confluence.atlassian.com/display/JIRA/Adding+the+Activity+Stream+Gadget
194 https://confluence.atlassian.com/display/JIRA/Updating+JIRA+Plugins+for+JIRA+4.2?focusedCommentId=228623879#comment-228623879
195 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

6.5.2.1 The New Project Is Not Structure Enabled

Make sure the project where the issue was moved to is enabled for Structure(see page 954). Structure ignores issues in the projects that are not Structure-enabled, so the moved issue is ignored as well, as if it ceased to exist. If you need this issue in the structure, either include the project where the issue resides now into the list of Structure-enabled projects(see page 954) or move the issue to an already Structure-enabled project (such as the original project).

6.5.2.2 The Issue Was Removed Based on Automation Rules

If the structure was built using Generators(see page 140), make sure the issue's new location satisfies the rules of your Insert(see page 142) and/or Extend (see page 148)generators. If it does not, you will need to change your generator rules or move the issue.

6.6 User Cannot Access Structure

6.6.1 Question

The user cannot see the Structure menu or access any structures. How do I resolve this problem?

6.6.2 Answer

To access Structure:

1. Structure has to be enabled for the user(see page 955) - Check which groups the user belongs to and ensure at least one of them has Structure access.
2. Structure must be enabled in any project(see page 954) that the user has permissions to view - Check which projects the user has permissions to view in Jira and make sure they are enabled for Structure.

6.6.2.1 If the user's permissions were just changed:

Configured permissions related to Structure are cached on the server, so for a couple of minutes after permissions are changed the user may not be able to access Structure. These caches will last for approximately 5 minutes before they automatically refresh. After that, the user should be able to use Structure.

To force a cache refresh, the user should do a *hard refresh* of their browser on any Jira page. After that, they should be able to use Structure immediately. In most browsers, hard refresh is achieved by clicking the Refresh button while holding the `Ctrl` or `Shift` button. If that doesn't work, check this list of ways to do a hard refresh in all popular browsers on Wikipedia: http://en.wikipedia.org/wiki/Wikipedia:Bypass_your_cache.

6.7 Where Do I Find the Jira Server ID?

Structure license is tied to a particular Jira Server, and a Server ID is required to generate a Structure license.

The Jira administrator can look up the 16-digit Server ID under **Administration | System Info** or **Administration | Structure | License Details**.

6.8 Using Subtasks and Structure

6.8.1 Question

Should I disable sub-tasks to use Structure?

6.8.2 Answer

Not necessarily. While Structure can be a good replacement for sub-tasks, they can be used in parallel — for example, if you want to try Structure on a single project without affecting other Jira users.

Structure treats sub-tasks as any other issues. If you're using Automation, you can also include a Sub-Tasks Extender(see page 152) to automatically add Jira sub-tasks beneath their parent issues.

6.9 Difference from Sub-tasks

6.9.1 Question

How is issue hierarchy provided by Structure different from the standard sub-tasks?

6.9.2 Answer

Sub-tasks have several major limitations:

- Sub-tasks are only a one-level hierarchy
- Sub-tasks are separate issue types
- Sub-tasks always inherit project and security level from their parent task

None of these limitations are present in Structure. At the same time, Structure provides all the features that sub-tasks have, and more. You can even use existing sub-tasks to build your structure's hierarchy, using the Sub-tasks Extender(see page 152).

6.10 Performance Considerations

If you have a large Jira instance (hundreds of thousands of issues), there are a few things to bear in mind when working with Structure.

- The recommended limit for the number of issues in one structure is 100K. Even at this point, Structure already might be working noticeably slower, especially if there are many users working with the Structure Board at the same time. If you have more issues than that, we recommend that you distribute the issues between several smaller structures (5-10K issues per structure works well). Having a large number of structures has less of an impact on performance.
- Structure's automation feature can place an additional load on servers, particularly as users are first learning to optimize their structures. Structure has some built-in protections, such as pausing Automation(see page 201) when generation times take too long, but you can also limit the access to Automation by changing permission to access Automation(see page 956).

- If necessary, you can switch off some parts of Structure(see page 973) to reduce the load (for example, the Structure panel on the Issue Page) and limit the group of users Structure is exposed to (see Structure Best Practices(see page 921)).
- If you experience other performance issues, please contact ALM Works Support[196].

6.11 How to restore the structure using History

There may be times when you need to restore a structure to some previous state. You can do so using the History panel.

To restore a structure using History:

1. Delete all current items in the structure before restoring its previous version from the history.
2. Open the structure History(see page 556) panel.
3. In the history, find and select the moment when the structure was in the desired state (before the unwanted changes took place).
4. Press CTRL+A (⌘+A on Macs) to select all issues and press CTRL+C (⌘+C) to cut them to clipboard.
5. Switch off history panel and press CTRL+V (⌘+V) – this should rearrange the structure according to the view you selected in the history.

> If you have some complicated synchronizers(see page 983) (for example, ones which use S-JQL in their configuration), we recommend temporarily disabling the synchronizers before restoring. After the restore is finished, you can enable them and run the resync.

6.12 Can a Structure be Exported to Microsoft Word?

Exporting to Microsoft Word is available through our integration with Xporter[197].

If you do not have Xporter installed, the following built-in export options are available:

- Export to Excel (see page 558)- Export your hierarchies into a fully-functional MS Excel spreadsheet.
- Export to Printable Page(see page 561) - In addition to creating a printer-friendly version of the structure, you can also use turn this into a shareable PDF file using your browser's Print to PDF option.

6.13 Convert time data in Excel export to Jira format

When you export a structure to Excel, the time tracking information is shown in hours format, rather than Jira's duration format (e.g., *1w 3d 5h 20m*).

To convert this data back to the Jira Duration format, use the Excel Macro and instructions below.

> Only Excel 2010+ is supported. It is possible that the macro will work with the 2007 version, but we can't guarantee it.

196 mailto:support@almworks.com
197 https://www.getxporter.app/blog/xporter-integration-with-structure/

6.13.1 How to Install

1. Download the attached Macro: FormatTimetrackingData.xlam[198].
2. In Excel, open **File** | **Options** | **Add-Ins.**
    a. Select **Manage** | **Excel Add-Ins** and click **Go**.
    b. In the dialog, click the 'Browse' button and select the downloaded `FormatTimetrackingData.xlam`
    c. Make sure the checkbox is selected for the 'FormatTimetrackingData' option and click **OK**.
3. In Excel, go to **File** | **Options** | **Customize Ribbon.**
    a. On the right, select the tab where the button for running the macro will go (e.g., *View).* Select the desired tab.
    b. Click **New Group** to add a custom group in the selected tab.
    c. Select the added group and click **Rename**. Select an icon and a name for it and click **OK**. A button for using the macro will be created.
    d. Go to **Choose commands from** | **Macros**, select FormatTrackingData, click **Add** and **OK**.

6.13.2 How to Use

1. Select the data to convert.
2. Click the created button on the ribbon.

The selected cells will be converted to a string in the Jira duration format, such as "1w 3d 5h 20m".

6.14 Data Center Approved Apps FAQ

Atlassian has established a class of marketplace apps built for Data Center. These Data Center approved apps must adhere to development requirements and undergo additional testing to ensure they meet the unique requirements of large-scare Data Center environments. If you are currently running the Structure for Jira server app on Data Center, you will need to upgrade to the Data Center version in order to continue receiving support.

The following Frequently Asked Questions will help explain these changes, as well as the timelines Atlassian has set for transitioning to Data Center approved apps.

6.14.1 What are Data Center approved apps?

Atlassian has established development and testing criteria for Marketplace apps used in Data Center environments. These include elements of how apps handle cache operations, support required databases, implement locking and availability in clustered environments, manage event handlers and much more.

These standards mean when you purchase a Data Center approved app, you can be confident it will meet the high demands of your large-scale Data Center environment. In order to be listed as a Data Center approved app, developers must present evidence that they have met these new standards and their testing results must be approved by Atlassian.

198 https://wiki.almworks.com/download/attachments/62756887/FormatTimetrackingData.xlam?api=v2&modificationDate=1549373146000&version=1

To learn more about the importance of using Data Center approved apps, see the Atlassian Data Center FAQ[199].

6.14.2 What are the criteria for being Data Center approved?

In order to be listed as a Data Center approved app, developers must adhere to rigorous development and testing criteria, designed to ensure such apps will perform to the unique requirements of large-scale Data Center environments. These addition tests focus on the unique needs of Data Center instances, such as caching, database support, clustered environments, and more.

In order to be considered for Data Center approval, developers must:

1. Complete a 100+ question readiness assessment, including an architectural review
2. Complete additional testing to evaluate their apps in an environment that simulates a large-scale Data Center instance

Once these additional steps are completed, Atlassian evaluates the readiness of each app and makes the final decision of whether an app can be designated Data Center approved.

To learn more about the Data Center approved app criteria, see Developing Apps for Atlassian Data Center Products[200].

199 https://www.atlassian.com/licensing/data-center-approved-apps
200 https://developer.atlassian.com/platform/marketplace/developing-apps-for-atlassian-data-center-products/

7 Release Notes

Find out about our latest updates.

7.1 Structure 8.3 Release Notes

26th of April 2023

Structure 8.3 introduces the Tempo Work Logged column, formula debugging tool, and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[201] (no installation required)

7.1.1 Version Highlights

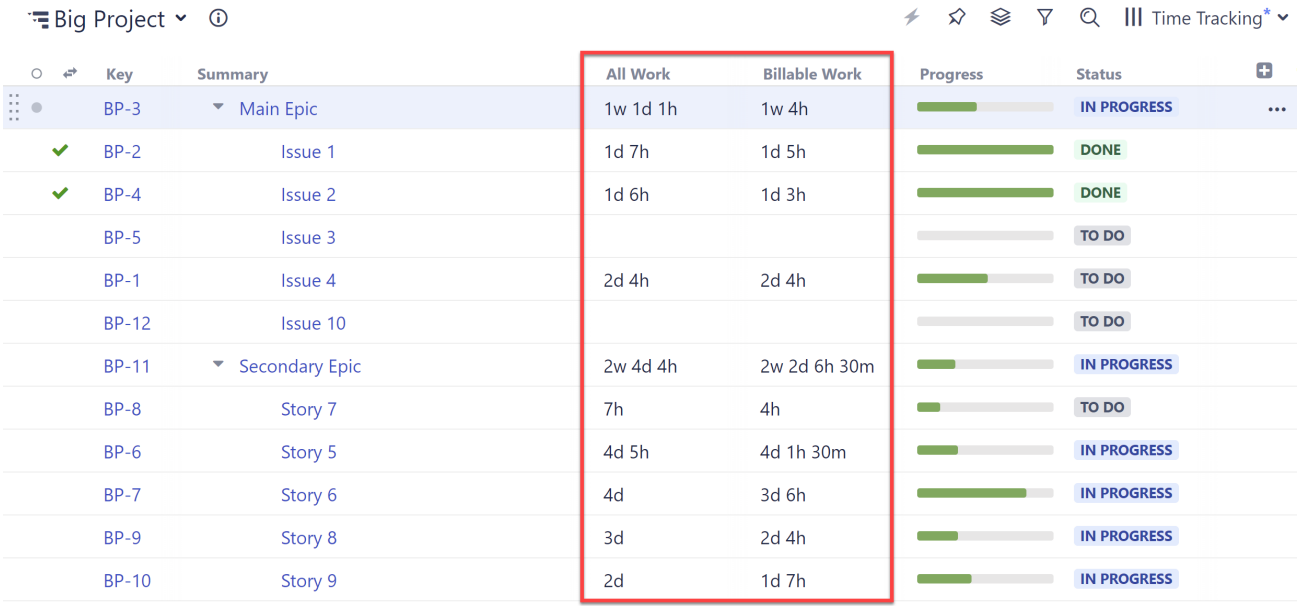
- Tempo Work Logged column and formulas integration
- Formula debugging tool
- Effector conflict resolution

7.1.2 Changes in Detail

7.1.2.1 Tempo Work Logged Column and Formulas Integration

The new Tempo Work Logged column allows you to roll up the time logged and billed for any project and reliably report on its progress:

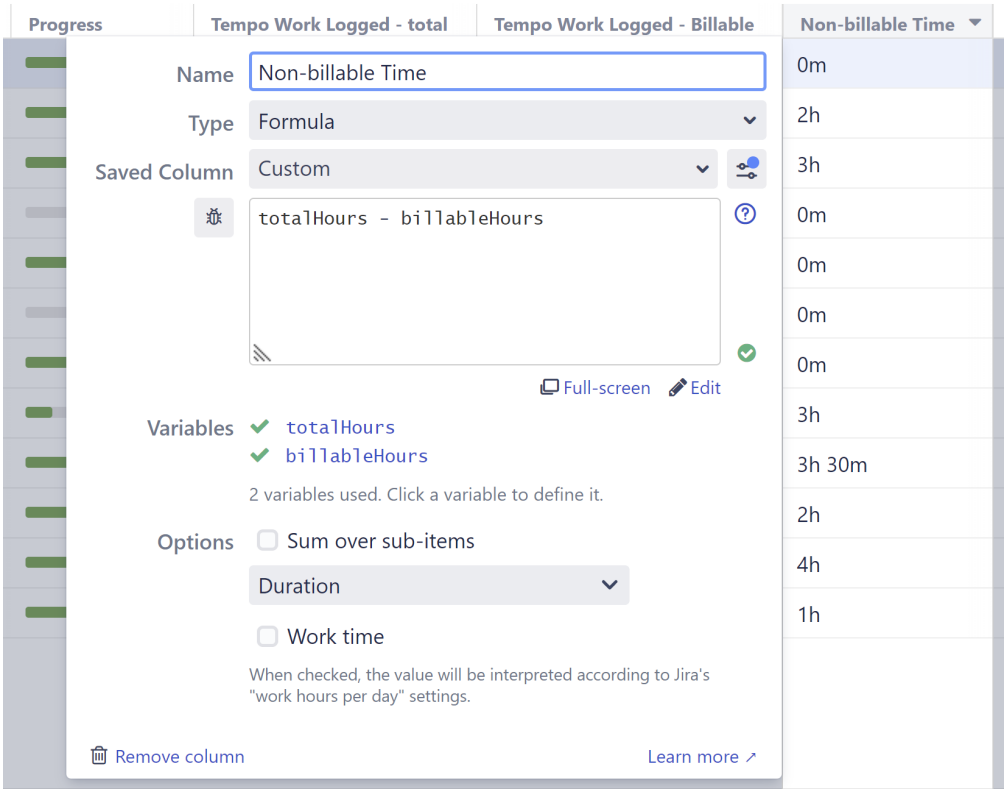
- Track logged and billable time alongside your project data
- Aggregate the time spent across any project hierarchy
- Create multiple columns to track specific users, time frames, billable/non-billable time, and more



Key	Summary	All Work	Billable Work	Progress	Status
BP-3	Main Epic	1w 1d 1h	1w 4h		IN PROGRESS
BP-2	Issue 1	1d 7h	1d 5h		DONE
BP-4	Issue 2	1d 6h	1d 3h		DONE
BP-5	Issue 3				TO DO
BP-1	Issue 4	2d 4h	2d 4h		TO DO
BP-12	Issue 10				TO DO
BP-11	Secondary Epic	2w 4d 4h	2w 2d 6h 30m		IN PROGRESS
BP-8	Story 7	7h	4h		TO DO
BP-6	Story 5	4d 5h	4d 1h 30m		IN PROGRESS
BP-7	Story 6	4d	3d 6h		IN PROGRESS
BP-9	Story 8	3d	2d 4h		IN PROGRESS
BP-10	Story 9	2d	1d 7h		IN PROGRESS

Tempo Work Logged data can also be used within formulas to do calculations or comparisons based on your work logged values.

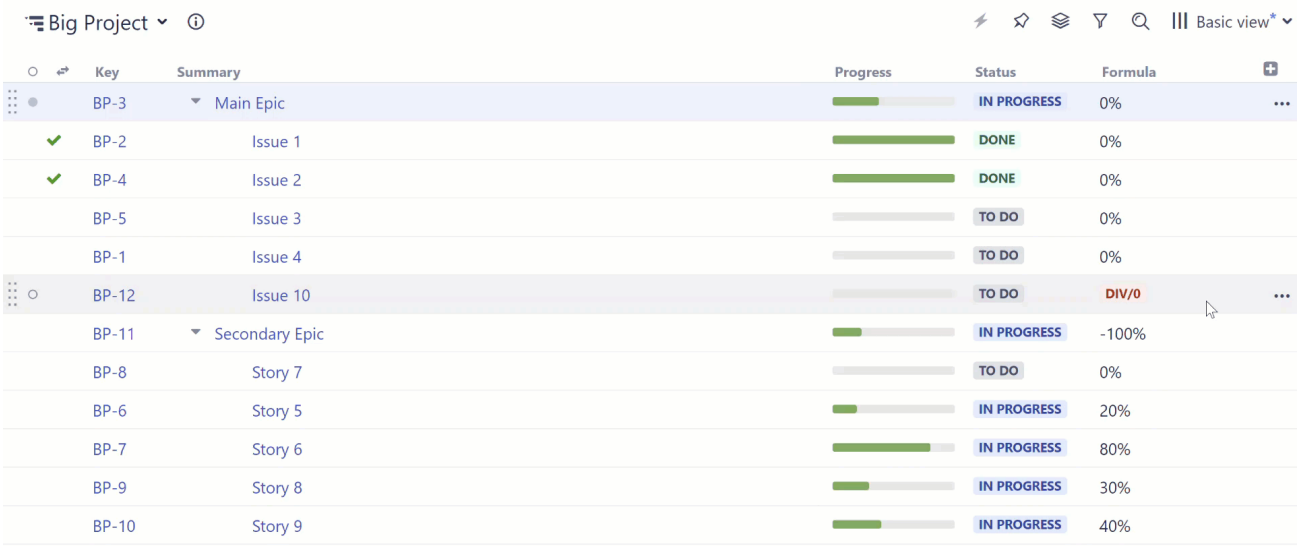
201 https://demo-structure.almworks.com/



Documentation: Tempo Work Logged Column(see page 491), Timesheets by Tempo(see page 600)

7.1.2.2 Formula Debugging Tool

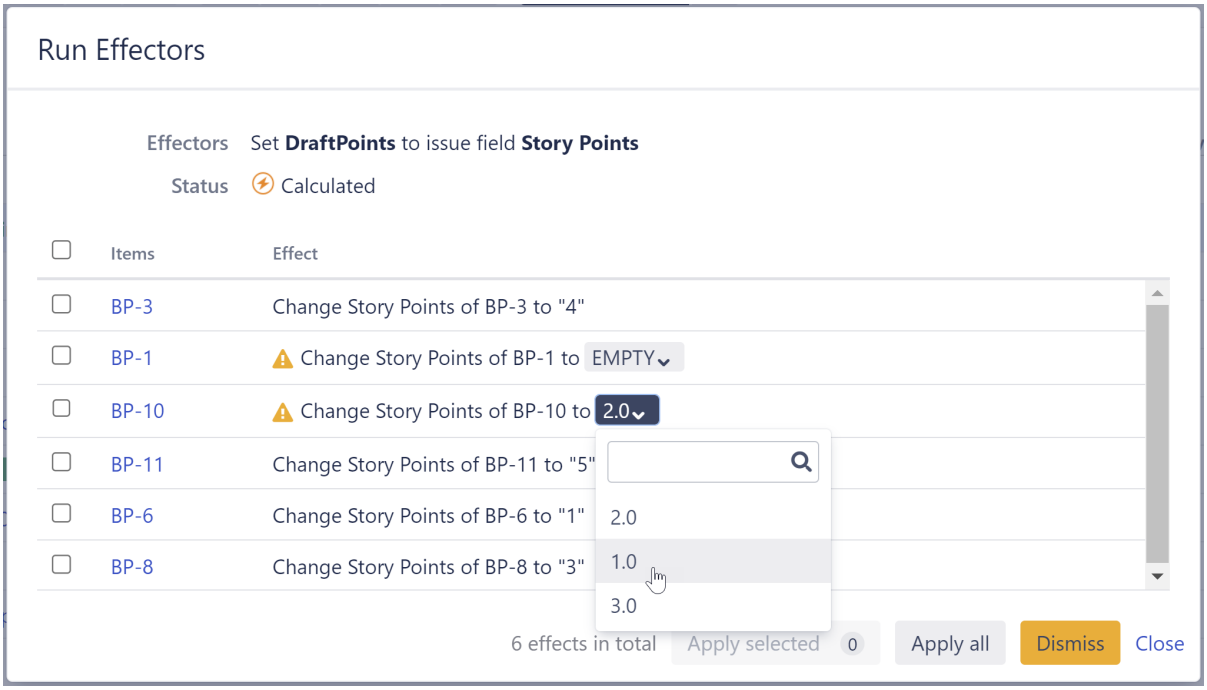
Tracking down problems with a formula just got a lot easier - with the new formula debugging tool, you can check the value of any variable in a formula for any row in the structure.



Documentation: Debugging a Formula(see page 262)

7.1.2.3 Effector Conflict Resolution

When running an Effector, if an issue has more than one value for the source attribute, you can now select the value or values you want to write to the Jira field.



Documentation: Running an Effector(see page 186)

7.1.2.4 Additional Improvements

- Filter by Field Generator(see page 167) improvements
- Formulas: Logarithm function support - LN(see page 0), LOG(see page 0), LOG10(see page 0)

7.1.3 Supported Versions

Structure 8.3 and all its extensions **support Jira versions 8.20 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

- Structure.Gantt[202] 3.0+
- Structure.Pages[203] 1.6+
- Structure.Testy [204]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

202 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
203 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
204 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.1.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.1.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [205] page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.1.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–8.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.1.5 Enterprise Deployment Notes

There are no special tests needed, so we suggest the usual testing procedures you've done for previous upgrades.

> Need help or have questions? Contact Tempo Support[206].

7.1.6 API Changes in Structure 8.3

7.1.6.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

---

205 https://wiki.almworks.com/display/structure2gmaster/Download
206 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Jira Version	New API Version
8.20+	17.18.0

See Structure API Versions[207] for full version information and downloads.

7.1.6.2 2. Compatible Changes in the Java API

`EffectorProcessManager.confirm(long)` method has been added to start applying all effects for a given process with a calculated preview.

7.1.7 Structure 8.3.1 Release Notes

> **23rd of May 2023**
>
> Structure 8.3.1 fixes issues with the linked issues section, Filter by Field, and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[208] (no installation required)

7.1.7.1 Version Highlights

- Fixed: Linked issues section sometimes collapses on the issue details panel after expanding
- Fixed: Only the first 100 sprints are available in the Filter by Field dropdown
- Fixed: Synchronizer is not migrated if user has a different key on the target Jira
- Fixed: Jira 9.0 compatibility, XSRF Token missing on bulk edit

7.1.7.2 Supported Versions

Structure 8.2 and all its extensions **support Jira versions 8.20 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

- Structure.Gantt[209] 3.0+
- Structure.Pages[210] 1.6+
- Structure.Testy [211]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

207 https://wiki.almworks.com/display/structure/.Structure+API+Versions+vFUTURE-dc
208 https://demo-structure.almworks.com/
209 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
210 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
211 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.1.7.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [212]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–8.3 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.1.7.4 Enterprise Deployment Notes

There are no special tests needed, so we suggest the usual testing procedures you've done for previous upgrades.

> Need help or have questions? Contact Tempo Support[213].

7.2 Structure 8.2 Release Notes

> **15th of December 2022**
>
> Structure 8.2 introduces Filter by Field and the ability to save and share formulas

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[214] (no installation required)

212 https://wiki.almworks.com/display/structure2gmaster/Download
213 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
214 https://demo-structure.almworks.com/

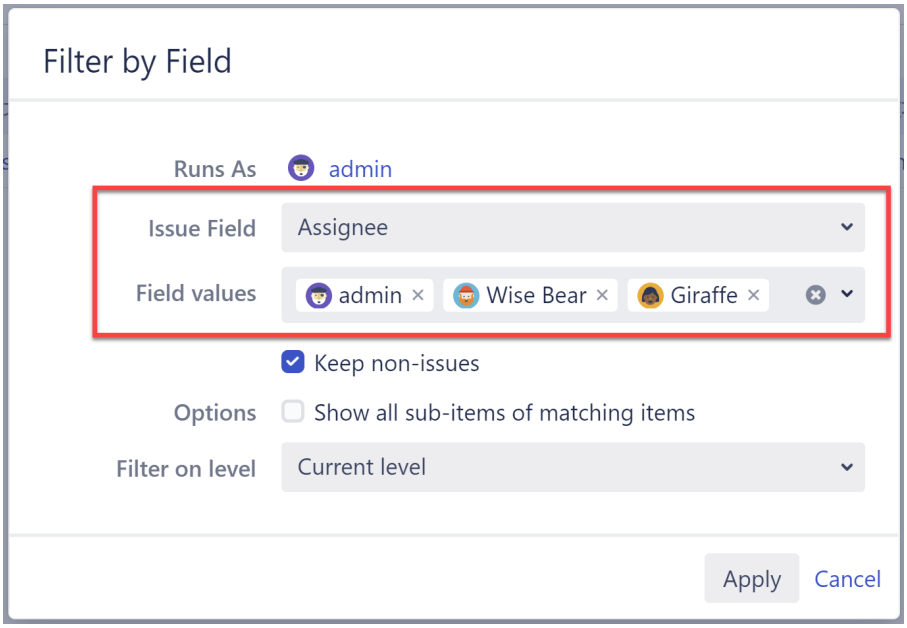
7.2.1 Version Highlights

- New filter: Filter by field
- Save and share formulas
- Manage Structure page improvements: Last opening date and new UI for configuration (note: last opening date will be empty for old structures)

7.2.2 Changes in Detail

7.2.2.1 Filter by Field

Filter by field is the easiest way to apply a simple filter to the structure, based on the values in a Jira field.

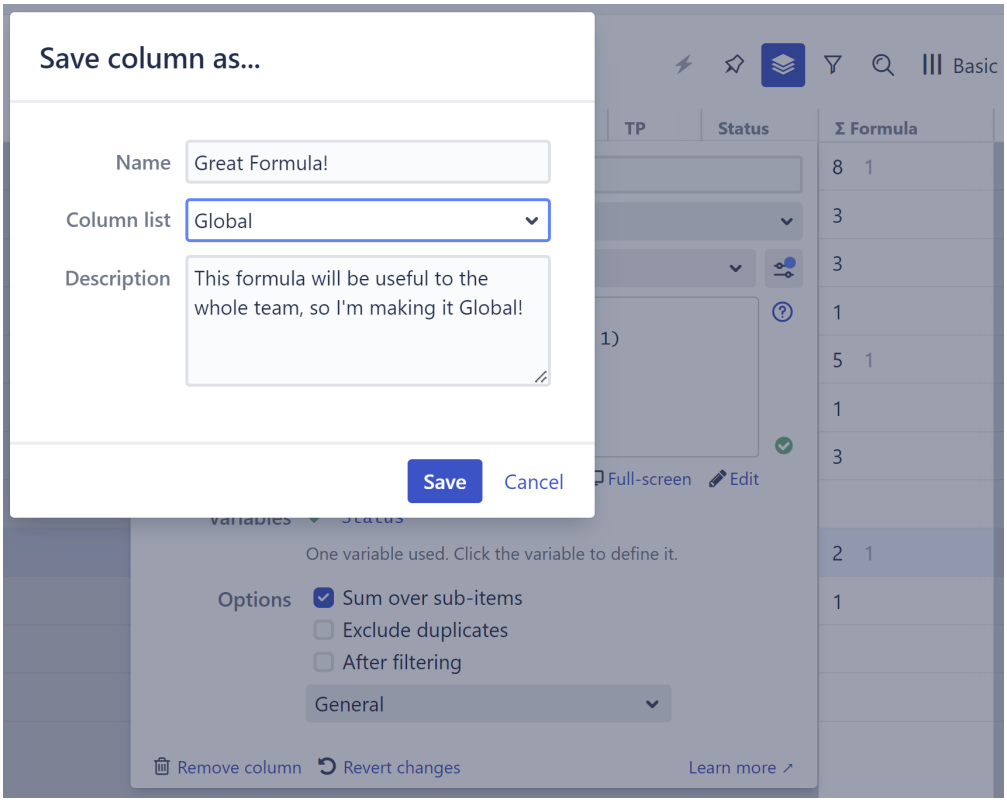


Filter by Field is available as a Transformation or Generator.

Documentation: Filter by Field Transformation(see page 213), Filter by Field Generator(see page 167)

7.2.2.2 Save and Share Formulas

It's now possible to save your formula columns to reuse in another structure, share with others, and even use in generators! If you have a great formula you want to use again and again, save it to your personal Saved Columns list, or add it to the Global list to share with your entire time.



Documentation: Saving a Formula(see page 260), Saved Columns(see page 524)

7.2.2.3 Additional Improvements

- Support for displaying name in the user() function
- Effectors now support cascading fields: not just "parent", but sub-levels, too
- Calendar option has been added to the Time in Status attribute (so it can be easily used in all formulas)
- Welcome screen: the new "[" hotkey opens and closes the admin tips toolbar (help section)
- Notification if you exceed the limit of an Insert generator and there are additional results
- Structure Gadget: it's now possible to edit text fields(multiline) and fields provided by the Custom Fields + +[215] App
- Security fixes
- Fixed: Effector failed to choose the option in a target field with Select List type
- Fixed: Notes were not copied for memos and folders when copying a structure

7.2.3 Supported Versions

Structure 8.2 and all its extensions **support Jira versions 8.13 or later**. **This is the last version with Jira 8.13 support**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

- Structure.Gantt[216] 3.0+

215 https://marketplace.atlassian.com/apps/1216682/custom-fields-for-jira?hosting=server&tab=overview
216 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

- Structure.Pages[217] 1.6+
- Structure.Testy [218]2.6+

Cloud customers can learn more about our products on the “Cloud” tab of our marketplace listing.

7.2.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.2.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [219]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.2.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–8.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.2.5 Enterprise Deployment Notes

There are no special tests needed, so we suggest the usual testing procedures you’ve done for previous upgrades. Note, there is no Saved Columns functionality in Cloud, so columns created after the update will not migrate correctly at this time.

> Need help or have questions? Contact Tempo Support[220].

217 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
218 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
219 https://wiki.almworks.com/display/structure2gmaster/Download
220 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.2.6 API Changes in Structure 8.2

7.2.6.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
8.13+	17.16.0

See Structure API Versions[221] for full version information and downloads.

7.2.6.2 2. Compatible Changes in the Java API

We have introduced a permission for managing global saved columns: `CoreAppPermissions.MANAGE_GLOBAL_SAVED_COLUMNS`. See 2023-06-05_18-05-46_.Managing Global Saved Columns vFUTURE-dc(see page 691) for details.

Also there are new structure errors related to saved columns:

- `StructureErrors.SAVED_COLUMN_NOT_EXISTS_OR_NOT_ACCESSIBLE`
- `StructureErrors.SAVED_COLUMN_MANAGEMENT_DENIED`

`BackupOperation` has a new `setBackupSavedColumns(boolean)` method to include saved columns in backup.

7.2.7 Structure 8.2.1 Release Notes

> **9th of January 2022**
>
> Structure 8.2.1 fixes context support in Filter by Field, Jira 9.5 support and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[222] (no installation required)

7.2.7.1 Version Highlights

- Added: compatibility with other Structure apps for Jira Server/Data Center
- Fixed: support multiple contexts in Filter by Field
- Fixed: Jira hotkeys on the Manage Saved Columns page
- Fixed: it was not possible to enter a date in the work logged column in Jira 9.5
- Fixed: always use timezone of column creator in Saved Columns
- Fixed: My Columns cannot be shared with a perspective link
- Minor security fixes

221 https://wiki.almworks.com/display/structure/.Structure+API+Versions+vFUTURE-dc
222 https://demo-structure.almworks.com/

7.2.7.2 Supported Versions

Structure 8.2 and all its extensions **support Jira versions 8.13 or later**. **This is the last version with Jira 8.13 support**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

- Structure.Gantt[223] 3.0+
- Structure.Pages[224] 1.6+
- Structure.Testy [225]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.2.7.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [226]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–8.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.2.7.4 Enterprise Deployment Notes

There are no special tests needed, so we suggest the usual testing procedures you've done for previous upgrades.

---

223 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
224 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
225 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
226 https://wiki.almworks.com/display/structure2gmaster/Download

Need help or have questions? Contact Tempo Support[227].

7.2.8 Structure 8.2.2 Release Notes

26th of January 2023

Structure 8.2.2 fixes formula columns in Confluence and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[228] (no installation required)

7.2.8.1 Version Highlights

- Fixed: Completed issues were shown in the Backlog group
- Fixed: Impossible to download the S-JQL troubleshooting file on Jira 9.5
- Fixed: Adding formula columns from Confluence gadget

7.2.8.2 Supported Versions

Structure 8.2 and all its extensions **support Jira versions 8.13 or later**. **This is the last version with Jira 8.13 support**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

- Structure.Gantt[229] 3.0+
- Structure.Pages[230] 1.6+
- Structure.Testy [231]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.2.8.3 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

227 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
228 https://demo-structure.almworks.com/
229 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
230 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
231 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [232]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–8.2.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.2.8.4 Enterprise Deployment Notes

There are no special tests needed, so we suggest the usual testing procedures you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[233].

7.3 Structure 8.1 Release Notes

22th of August 2022

Structure 8.1 introduces new bundled formulas, project lead data, Help tool, and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[234] (no installation required)

7.3.1 Version Highlights

- Easier access to bundled formulas
- New bundled formulas
- Calendar support for the Time in Status Column
- Project Lead data for project groups
- Welcome to Structure page

232 https://wiki.almworks.com/display/structure2gmaster/Download
233 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
234 https://demo-structure.almworks.com/

- Additional improvements and fixes

7.3.2 Changes in Detail

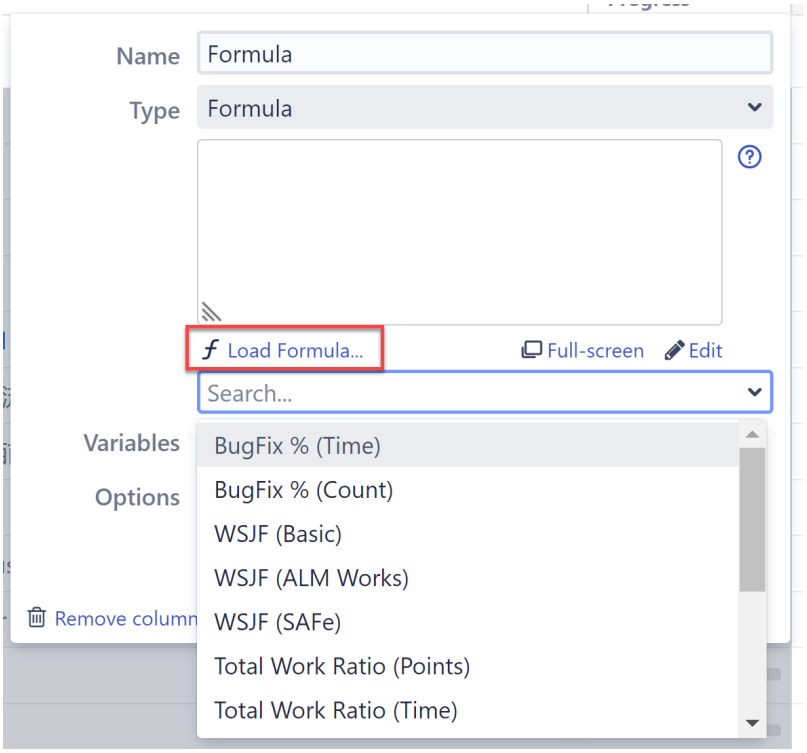
7.3.2.1 New Bundled Formulas

We've added 3 new bundled formulas:

- Issue Health - Provides a visual health status for each item, based on issue progress by time tracking
- Tasks Since Sprint Started - Calculates the number of tasks added since the last sprint began
- Time Flagged - Calculates the total time a task was flagged

7.3.2.2 Easier Access to Bundled Formulas

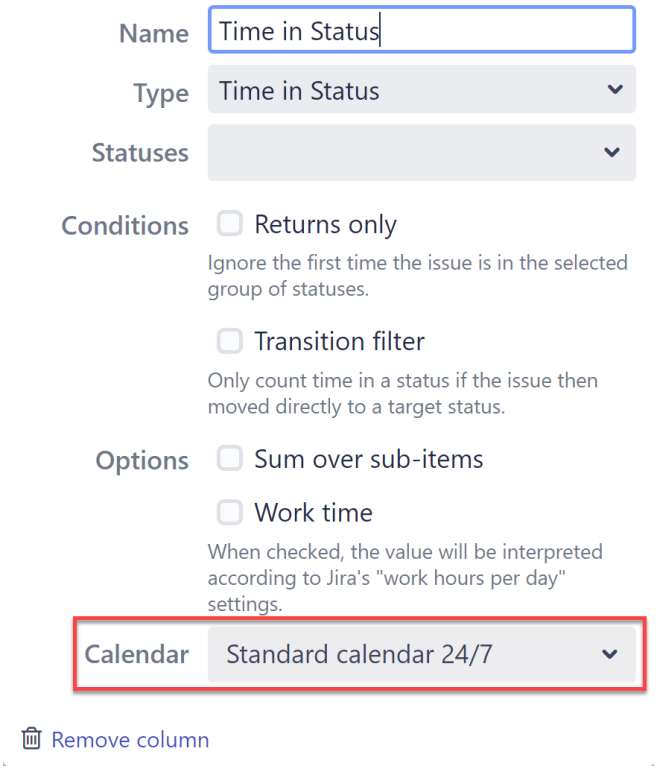
It's now even easier to try one of our bundled formulas - just add a Formula column(see page 229), click **Load Formula...**, and select the one you need.



As you select one of the bundled formulas, the results will be populated automatically. If it's not the one you're looking for, just select another from the list - the previous formula will be cleared, and the new one will take its place.

7.3.2.3 Calendar Support within the Time in Status Column

The Time in Status column now supports work calendars, which allow you to specify whether the Time in Status calculation should consider 24 hours a day, 7 days a week, or just working hours (8 hours, 5 days).

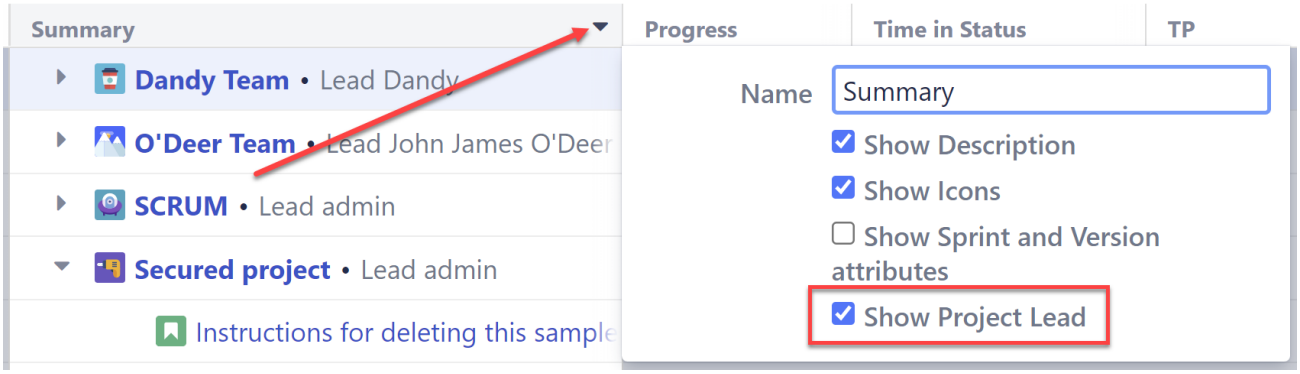


If you have Structure.Gantt[235]installed, you can also select from your custom Gantt calendars, to create custom results based on your specific work calendar, holidays, vacations, etc.

Documentation: Time in Status Column(see page 493)

7.3.2.4 Project Lead Data

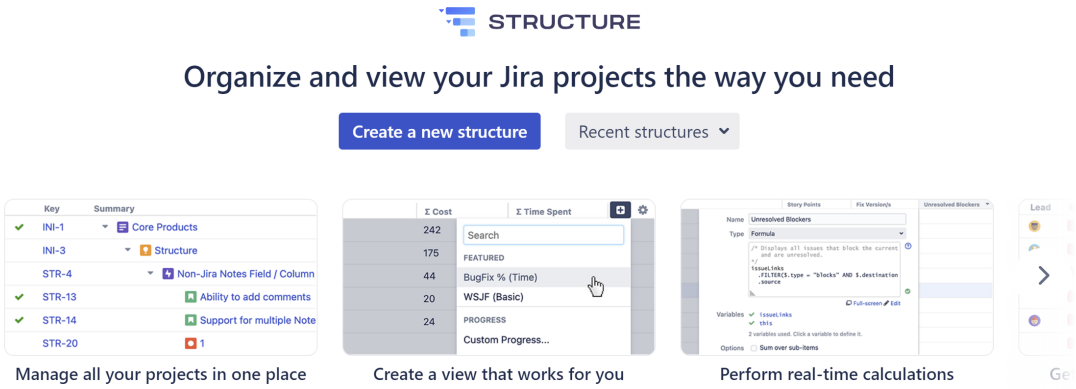
Not sure who to contact about a project? If you've grouped your issues by Project, either using a Group generator or transformation, you can now display the Project Lead in the summary column.



235 https://marketplace.atlassian.com/apps/1217809/structure-gantt-gantt-charts-roadmaps-for-jira?hosting=datacenter&tab=overview

7.3.2.5 Welcome to Structure Page and Help Tool

We've added a new "Welcome to Structure" page to make it easier for teams to get the most out of Structure. Quickly discover the most popular benefits, learn how to build structures for many popular use cases, and get help when you need it!

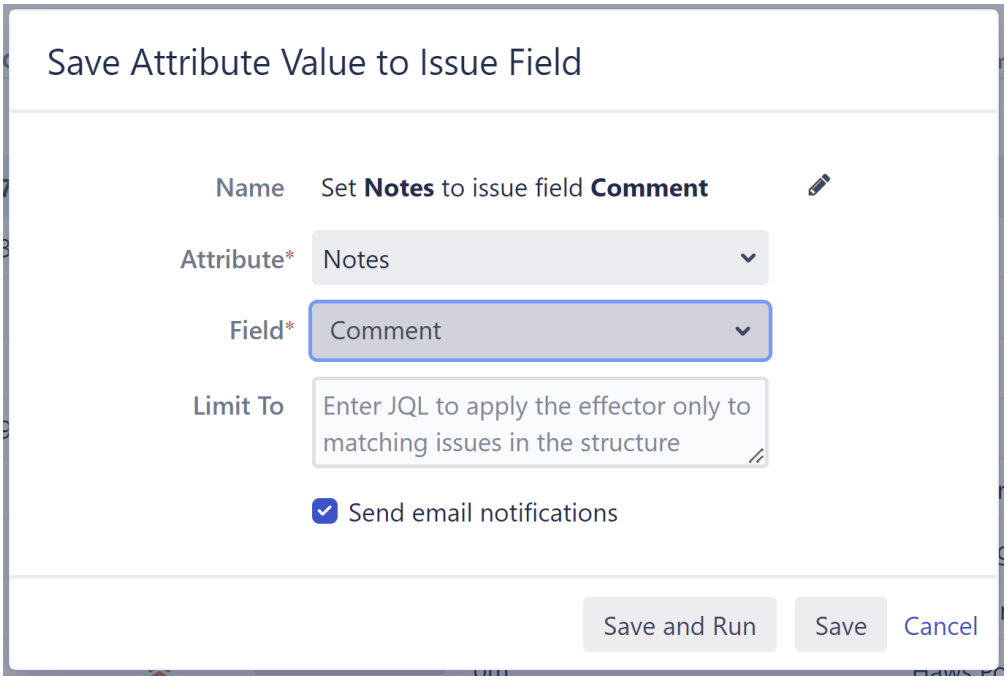


Jira admins can also add a custom Help tool to the Welcome page, where you can add team-specific tips, include helpful links, and provide contact information for your support team.

Documentation: Help Tool(see page 1027)

7.3.2.6 Write to the Comments Field with Effectors

The Attribute to Field Effector now supports writing to the Comments field. Call attention to your Structure notes, the results of a formula, or any other data you think everyone should see, by making them the latest comment for each issue.



Documentation: Attribute to Issue Field Effector(see page 179)

7.3.2.7 Additional Improvements

- Calendar_duration function to use calendars in formulas
- Fixed: "Allow changes via structure" option was hidden in the Insert Structure and Extend with Linked Issues generator settings
- Fixed: export to Excel contained html tags when text was created in JEditor
- Additional performance improvements and fixes

7.3.3 Supported Versions

Structure 8.1 and all its extensions **support Jira versions 8.13 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

- Structure.Gantt[236] 3.0+
- Structure.Pages[237] 1.6+
- Structure.Testy [238]2.6+

236 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
237 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
238 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

Cloud customers can learn more about our products on the “Cloud” tab of our marketplace listing.

7.3.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.3.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [239]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.3.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–8.0 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.3.5 Enterprise Deployment Notes

In this release we've fixed our mechanism for project reindex detection. There was a bug that caused frequent cache clearances (as a result of some Jira automation or app activity) and full recalculation of all structures in use. Now project reindex detection works more precisely.

There was also a small concurrency improvement implemented, so performance should be improved during high load hours.

There are no special tests needed, so we suggest the usual testing procedures you've done for previous upgrades.

> Need help or have questions? Contact Tempo Support[240].

239 https://wiki.almworks.com/display/structure2gmaster/Download
240 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.3.6 API Changes in Structure 8.1

7.3.6.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
8.13+	17.15.0

See Structure API Versions[241] for full version information and downloads.

7.3.6.2 2. Compatible Changes in the Java API

We have introduced SPI for defining custom work calendars. It is meant to be used by ALM Works products and third-party apps that integrate with Structure. Currently working calendars are only supported in Time in Status Column(see page 493). SPI interfaces can be found in a new `com.almworks.jira.structure.api.calendar` package.

7.3.7 Structure 8.1.1 Release Notes

> **30th of August 2022**
>
> Structure 8.1.1 is a patch release for Structure 8.1. It fixes an issue with using formulas as variables

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[242] (no installation required)

7.3.7.1 Version Highlights

Structure 8.1.1 is a patch release for Structure 8.1. It provides:

- Fixed: unable to use formulas as variables in other formulas
- Fixed: blank formula editor in the Personal Backlog view

7.3.7.2 Supported Versions

Structure 8.1 and all its extensions **support Jira versions 8.13 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

241 https://wiki.almworks.com/display/structure/.Structure+API+Versions+vFUTURE-dc
242 https://demo-structure.almworks.com/

- Structure.Gantt[243] 3.0+
- Structure.Pages[244] 1.6+
- Structure.Testy [245]2.6+

Cloud customers can learn more about our products on the “Cloud” tab of our marketplace listing.

7.3.7.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [246]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–8.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.3.7.4 Enterprise Deployment Notes

There are no special tests needed, so we suggest the usual testing procedures you've done for previous upgrades.

> Need help or have questions? Contact Tempo Support[247].

243 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
244 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
245 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
246 https://wiki.almworks.com/display/structure2gmaster/Download
247 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.3.8 Structure 8.1.2 Release Notes

13th of October 2022

Structure 8.1.2 is a patch release for Structure 8.1. It fixes an issue with JQL in the Personal Backlog template

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[248] (no installation required)

7.3.8.1 Version Highlights

Structure 8.1.2 is a patch release for Structure 8.1. It provides:

- Fixed: no issues in the Personal Backlog for new users

7.3.8.2 Supported Versions

Structure 8.2 and all its extensions **support Jira versions 8.13 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

- Structure.Gantt[249] 3.0+
- Structure.Pages[250] 1.6+
- Structure.Testy [251]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.3.8.3 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [252]page.

248 https://demo-structure.almworks.com/
249 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
250 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
251 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
252 https://wiki.almworks.com/display/structure2gmaster/Download

2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–8.1.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.3.8.4 Enterprise Deployment Notes

There are no special tests needed, so we suggest the usual testing procedures you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[253].

7.4 Structure 8.0 Release Notes

30th of March 2022

Structure 8.0 introduces text wrapping

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[254] (no installation required)

7.4.1 Version Highlights

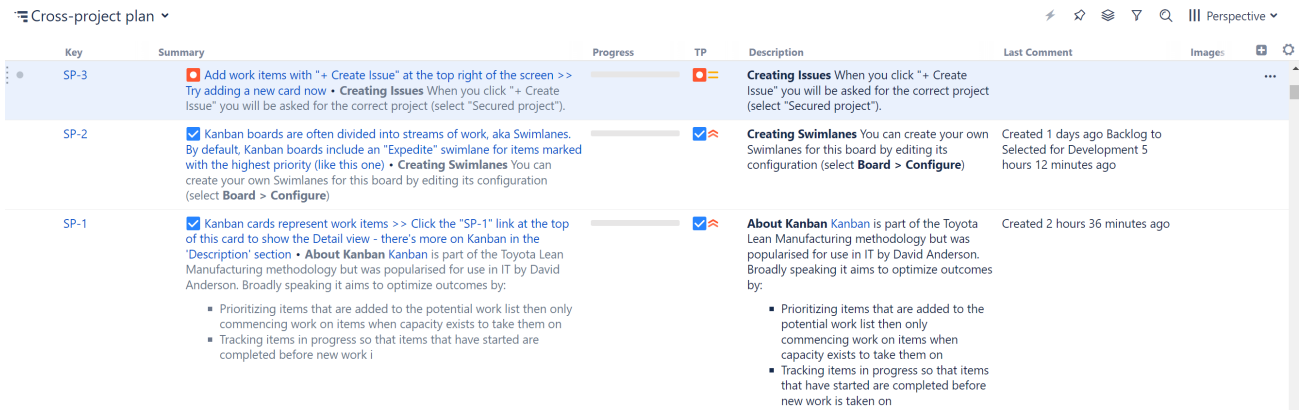
- Text wrapping
- Memo attributes in formulas
- Change a view owner

253 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
254 https://demo-structure.almworks.com/

7.4.2 Changes in Detail

7.4.2.1 Text Wrapping

See even more details about an issue at a glance! Text Wrapping expands each row to the height of the longest cell, so you can easily read longer text fields without opening the issue.



This was the #1 most requested feature on our Uservoice channel[255]- thank you to everyone who voted for it and please keep your feedback coming!

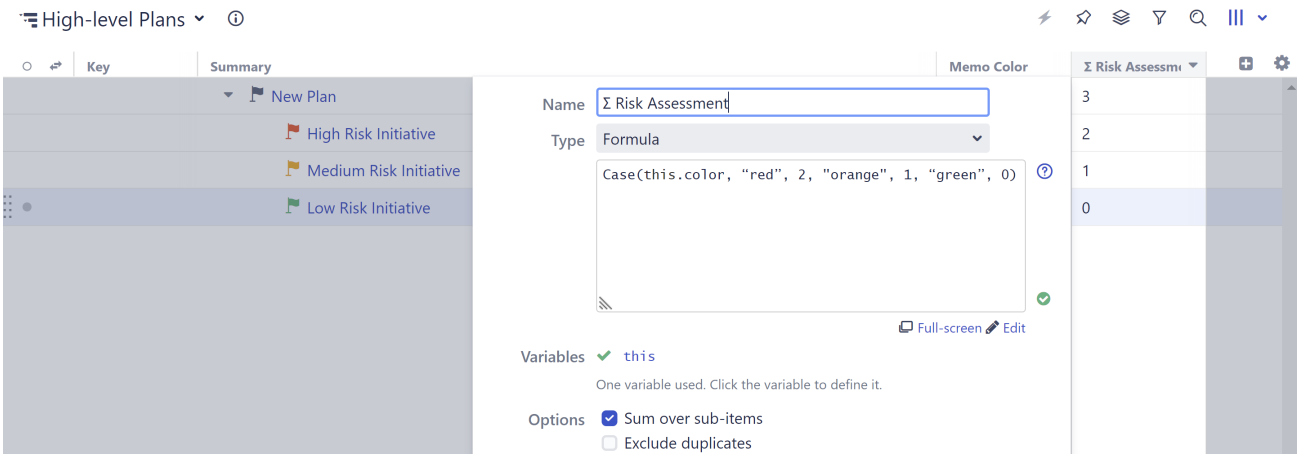
Documentation: Text Wrapping(see page 523)

7.4.2.2 Memo Attributes in Formulas

Memo's just got even more useful! You can now use a memo's color, description, and title as formula variables.

- Use memo colors to identify risk levels in your planning stage, and then quickly aggregate risk across the structure.
- Add key words to a memo's title or description, and narrow your formula results based on those.
- Group issues under memos, and create different calculation rules based on the memo values.

255 https://structure.uservoice.com/



You can can reference a memo's color, description, or title using the following formats:

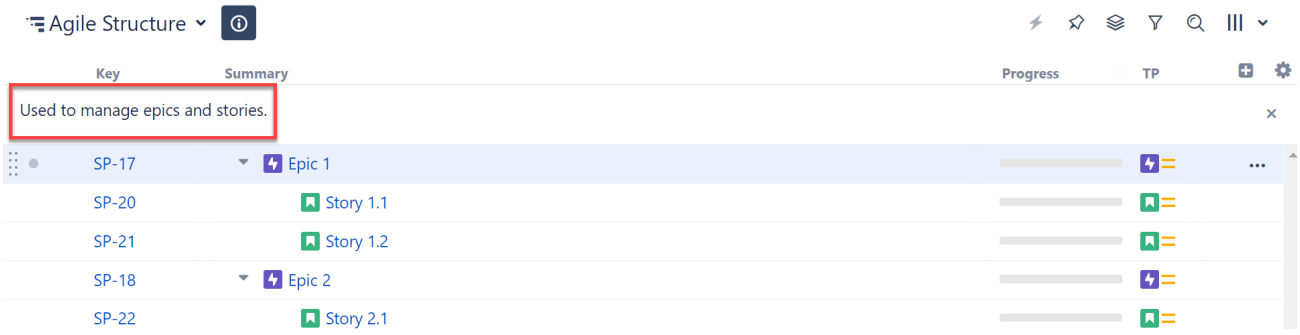
- this.color
- this.description
- this.title

> In the example above, we assigned a "risk" value to each memo color and aggregated those values across our plan.

Documenation: Memo(see page 105)

7.4.2.3 Structure Description

When you open a structure for the first time, its description is shown at the top of the Structure panel. This can be extremely useful when sharing a structure with someone else, as they can quickly see what its purpose is.



To close the description, or open it again, simply click the info button beside the structure's name

7.4.2.4 Additional Improvements

- Ability to change the owner of a view(see page 515)
- Enable Manual Adjustments(see page 194) by default

7.4.3 Supported Versions

Structure 8.0 and all its extensions **support Jira versions 8.13 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible with:

- Structure.Gantt[256] 3.0+
- Structure.Pages[257] 1.6+
- Structure.Testy [258]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.4.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.4.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [259]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.4.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–7.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.4.5 Enterprise Deployment Notes

Several changes were made in this release to improve application stability.

---

256 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
257 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
258 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
259 https://wiki.almworks.com/display/structure2gmaster/Download

7.4.5.1 Item Tracker Data Cleanup

We implemented a new maintenance task called "Clear item events table" to clean up cluster item events in Jira DC. We highly recommend this task be applied on instances where node renaming or new node addition is a frequent operation. The task can be scheduled periodically or run manually after a node operation. This task will remove all event information from Structure database tables to avoid event ID overflow. All used structures are recalculated as a side effect of this maintenance task, so it is highly recommended to run it in nonworking hours.

7.4.5.2 Attributes Loading Cache Limitations

In the Structure 7.4 release we introduced attribute calculation performance improvements through more efficient cache usage. Unfortunately, there was a mistake with the cache limit that can potentially cause *OutOfMemoryError* on exotic attribute configurations. This release fixes the cache limit. This may cause a slowdown in attributes calculation (aggregates or export). If you experience a problem, there is a dark feature that allows you to adjust the cache size for your instance to a reasonable limit (please contact Tempo Support[260] for help).

7.4.5.3 Structure Gadget in Confluence Stability Fix

We introduced a fix to the Confluence gadget when using Jira DC, so that some Structure attributes now have to be calculated by forest specification w/o filters (for instance, *Sequential Index*). If you use such attributes in a gadget with filters, it can cause an *ArrayIndexOutOfBoundsException* on the server side when the gadget reconnects to a different Jira node. The change fixes this problem.

7.4.5.4 Testing on Staging Environment

Apart from the changes and suggestions above, along with the usual testing procedures you've done for previous upgrades, we advise comparing attribute load times against Structure 7.4 for huge structures with aggregate attribute columns, if you use any. We also advise comparing export (Printable or Excel) for huge structures against Structure 7.4, if you use any.

Need help or have questions? Contact Tempo Support[261].

7.4.6 API Changes in Structure 8.0

7.4.6.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

260 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

261 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Jira Version	New API Version
8.13+	17.13.0

See Structure API Versions[262] for full version information and downloads.

7.4.6.2 2. Compatible Changes in the Java API

We have added a new parameter named "row display mode" to the view specification, with getters and setters:

- `ViewSpecification.getRowDisplayMode()`
- `ViewSpecification.Builder.getRowDisplayMode()`
- `ViewSpecification.Builder.setRowDisplayMode(int)`

Supported values are stored as `int` constants in the new class `RowDisplayMode`:

- `ONE_LINE` means that the structure rows have fixed height and row content is displayed in one line
- `TEXT_WRAPPING` means that the structure rows have adaptive height to show full row content

The default row display mode is `ONE_LINE`.

7.4.7 Structure 8.0.1 Release Notes

25 of April 2022

Structure 8.0.1 is a patch release for 8.0. It fixes an issue with compatibility, views, Effectors, and more.

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[263] (no installation required)

7.4.7.1 Version Highlights

Structure 8.0.1 is a patch release for Structure 8.0. It provides:

- Fixed: compatibility with Draw.io
- Fixed: shared perspectives opened to a different location when text wrapping was switched on
- Fixed: view name did not appear on the Views menu dropdown in larger windows
- Fixed: Attribute to Field effector failed to choose the correct project for the Fix Version target field
- Adds compatibility with Jira 9 EAP

Upgrade from Structure 8.0 is highly recommended.

7.4.7.2 Supported Versions

Structure 8.0.1 and all its extensions **support Jira versions 8.13 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

---

262 https://wiki.almworks.com/display/structure/.Structure+API+Versions+vFUTURE-dc
263 https://demo-structure.almworks.com/

Compatible with:

- Structure.Gantt[264] 3.0+
- Structure.Pages[265] 1.6+
- Structure.Testy [266]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.4.7.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira app infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [267]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–8.0 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the app.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.4.7.4 Enterprise Deployment Notes

Testing on Staging Environment

There are no particular special areas of interest for load testing and stress testing Structure 8.0.1. We advise running the same testing procedures as you've done for previous upgrades.

---

264 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
265 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
266 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
267 https://wiki.almworks.com/display/structure2gmaster/Download

Need help or have questions? Contact Tempo Support[268].

7.5 Structure 7.4 Release Notes

21st of December 2021

Structure 7.4 lets you access historical field values in Formulas

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[269] (no installation required)

7.5.1 Version Highlights

- Access historical field values in formulas
- Fixed: Can't use work logged attribute in formulas
- Fixed: Aggregations are calculated incorrectly when called from an attribute group generator
- Small UI improvements: style and color changes to the history panel, time tracking, and automation banner
- Performance improvements

7.5.2 Changes in Detail

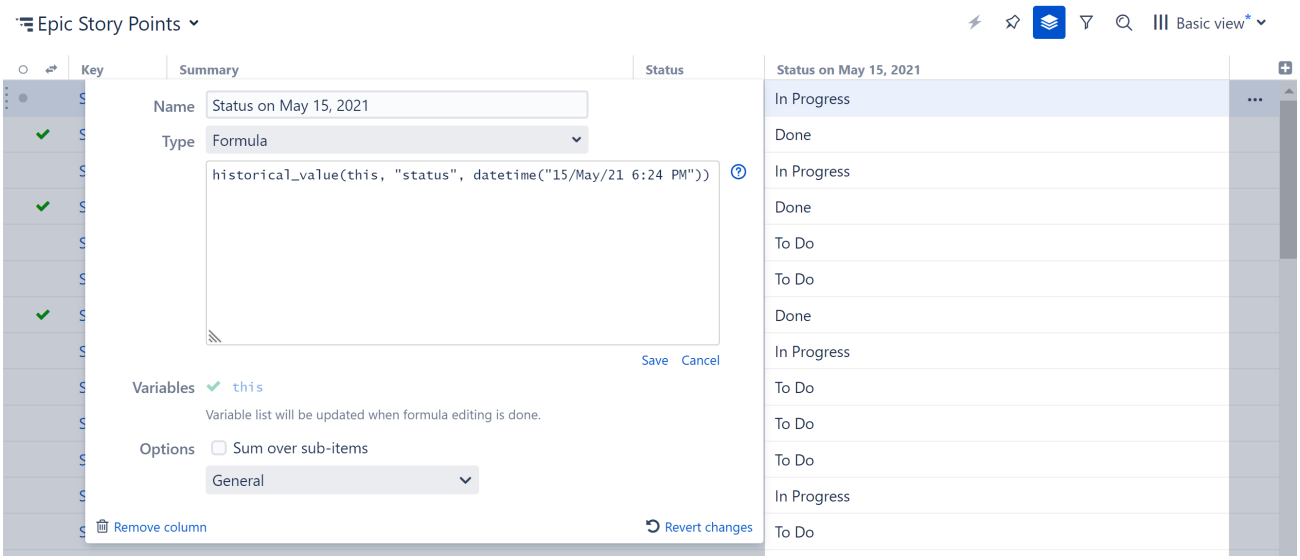
7.5.2.1 Historical Field Values in Formulas

Use formulas to tap into historical changes:

- Find out when an issue was last changed, who changed it, and what changes were made
- Easily prepare for quarterly results by pulling field values from a specific date - or compare values from different dates
- Count tasks added after a sprint began
- Calculate the planned value on a specific date (ex. What was the cost on September 1, 2019?)

268 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

269 https://demo-structure.almworks.com/



The example above is one of the simplest - but most useful - uses of historical values in formulas. You can do the same for any field and any date:

```
historical_value(this, "<FIELD_NAME>", datetime("<DATE_AND_TIME>"))
```

Documentation: Historical Value Function(see page 710), Sample Formulas for Historical Values(see page 0)

7.5.3 Supported Versions

Structure 7.4 and all extensions **support Jira versions 8.13 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[270] 3.0+
- Structure.Pages[271] 1.6+
- Structure.Testy [272]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.5.4 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

270 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
271 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
272 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.5.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [273]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.5.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–7.3 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.5.5 Enterprise Deployment Notes

7.5.5.1 Attributes caclulation improvement

We've included a number of performance improvements in this Christmas release, including changes to our attributes calculation routines, which helps Structure utilize cache more efficiently. As a result, aggregate calculations and export operations were up to 90% faster in our tests. We also improved caching for expr calculation algorithms, making heavy formulas with a lot of data involved, such those involving worklogs and comments, faster.

> The speed improvements for attribute and expr calculations will vary depending on your Jira setup and exact attribute and expression implementation.

7.5.5.2 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 7.4. We advise running the same testing procedures as you've done for previous upgrades.

> Need help or have questions? Contact Tempo Support[274].

273 https://wiki.almworks.com/display/structure2gmaster/Download
274 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.5.6 API Changes in Structure 7.4

7.5.6.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
8.13+	17.12.0

See Structure API Versions[275] for full version information and downloads.

7.5.6.2 2. Compatible Changes in the Java API

New change history item and change history group item types have been introduced in Structure 7.4. `CHANGE_HISTORY_ITEM` and `CHANGE_HISTORY_GROUP` constants have been added to `CoreItemTypes`. Also there are new methods in `CoreIdentities` to create corresponding item identities.

`BackupOperation.setBackupExtension(Set<String>)` method has been added to specify Structure extensions to backup along with the Structure data.

`GenerationContext.inputForest()` method has been added for getting the forest fragment generated before calling the current generator.

7.6 Structure 7.3 Release Notes

22nd of October, 2021

Structure 7.3 fixes Confluence gadget stability

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[276] (no installation required)

7.6.1 Version Highlights

- Fixed: a stability issue with the Structure gadget in Confluence, when making a request to Jira Data Center. More info(see page 715)
- Fixed: Query match column did not handle 'Sum over sub-items' properly
- Fixed: "attachments" variable wasn't automatically mapped in formula columns
- Fixed: broken UI when hovering over fields with wiki markup
- Fixed: Formula column with "Sum over sub-items" option selected could not be used as a variable

275 https://wiki.almworks.com/display/structure2gmaster/Structure+API+Versions
276 https://demo-structure.almworks.com/

- Fixed: exporting structures with complex formulas took significantly longer than expected
- Fixed: icons were not displaying in the Confluence gadget

7.6.2 Changes in Detail

7.6.2.1 Supported Versions

Structure 7.3 and all extensions **support Jira versions 8.5 or later**. **This is the latest release with 8.5 support, the next one will support Jira's 8.13+**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[277] 3.0+
- Structure.Pages[278] 1.6+
- Structure.Testy [279]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.6.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.6.3.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [280]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.6.3.2 Upgrading Structure

The upgrade procedure from versions 3.0–7.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.

277 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
278 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
279 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
280 https://wiki.almworks.com/display/structure2gmaster/Download

3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.6.4 Enterprise Deployment Notes

7.6.4.1 Structure Gadget in Confluence

This release introduces an update for Structure gadget in Confluence, when working with Jira Data Center. Previously, we discovered an issue with the session affinity mechanism, where any gadget request could be forwarded to any Jira Data Center node. Structure gadget would try a few times to reach the original node and then stop with an error banner after the failed retries. With this release, Structure gadget can now detect the node change and, after a few failed retries, reconnect to a new node and continue to work. Each reconnection works like a full update - all rows and attributes data will be loaded at once. If the node changes too often (it depends on the load balancer algorithm, Jira load, etc.) it can increase network traffic. There is a Dark Feature[281] that allows admins to switch back to the old behavior: *structure.gadget.remote.allowNodeWandering* - set to *false* to disable the node change behavior (*true* by default).

7.6.4.2 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 7.3. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[282].

7.6.5 API Changes in Structure 7.3

7.6.5.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible and affect only internal classes, so any code using API 17.x should work fine.

Jira Version	New API Version
8.5+	17.11.0

See Structure API Versions[283] for full version information and downloads.

281 https://wiki.almworks.com/display/structure/Advanced+Configuration+and+Dark+Features
282 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
283 https://wiki.almworks.com/display/structure2gmaster/Structure+API+Versions

7.7 Structure 7.2 Release Notes

31st of August, 2021

Structure 7.2 adds the ability to group Quick Transformations

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[284] (no installation required)

7.7.1 Version Highlights

Structure 7.2 allows you to group Quick Transformations, to apply multiple transformations at once.

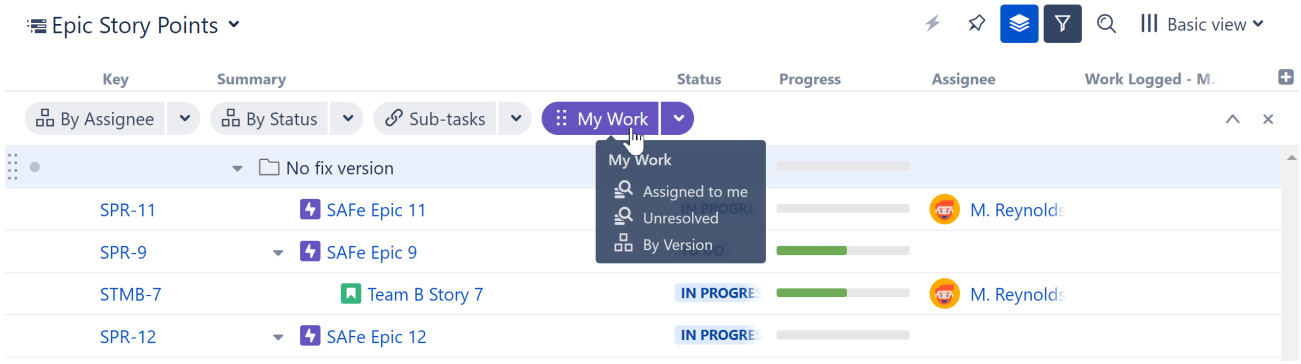
7.7.2 Changes in Detail

7.7.2.1 Transformation Groups

Apply several transformations with a single click.

Quick Transformations are a great way to temporarily refine a structure or change the way it's organized for a specific task - but often, you need to apply several transformations at the same time. For example, you may want to look at tasks assigned to you, that are unresolved, and grouped by Version...

Now you can apply all those transformations at once, by adding them to a group.



You can create as many Transformation groups as you need, each with a unique name and color, so they're easy to find and apply.

Documentation: Transformation Groups(see page 224)

7.7.2.2 Additional Updates

- Fixed: SUM{} didn't work with select lists
- Performance improvements for attribute calculation

284 https://demo-structure.almworks.com/

7.7.3 Supported Versions

Structure 7.2 and all extensions **support Jira versions 8.5 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[285] 2.7+
- Structure.Pages[286] 1.6+
- Structure.Testy [287]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.7.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.7.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [288]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.7.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–7.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

285 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
286 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
287 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
288 https://wiki.almworks.com/display/structure2gmaster/Download

7.7.5 Enterprise Deployment Notes

7.7.5.1 Attribute calculation performance improvement

In this release we improved the attribute calculation procedure. Structure now preloads all the issues required for attribute calculation with a bulk request. Actual numbers will depend on the structure configuration, amount of issues in the structure, aggregates usage, database performance, etc.; however, in our internal tests we saw a 20-100% speedup in attribute loading.

7.7.5.2 Downgrade to previous releases

Structure 7.2 introduces a new feature called Transformation Grouping. It allows you to create groups of transformations and disable and enable the whole group at once. This functionality changes the way transformations are organized in the database.

A special Groovy (ScriptRunner) script should be applied before or immediately after downgrading to ensure transformations are not lost. If applying the script after a downgrade, do not make any changes to your transformations before running the script, or transformation data may be lost.

The following script ungroups all quick transformation groups for all structures:

Migrate quick transforms

```
import groovy.json.JsonSlurper
import groovy.json.JsonOutput
import com.almworks.jira.structure.api.StructureComponents
import com.almworks.jira.structure.api.property.StructurePropertyService
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.onresolve.scriptrunner.runner.ScriptRunnerImpl

import static java.util.Arrays.asList;

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version =
'17.0.0')
@WithPlugin('com.almworks.jira.structure')

class QuickTransformsMigration {
  def log
  def sps

  void run() {
    def sc = ScriptRunnerImpl.getPluginComponent(StructureComponents)
    sps = sc.getStructurePropertyService()
    def sm = sc.getStructureManager()
    sm.getAllStructures(null).each { s ->
      migrate(s.id)
    }
    log.warn("Migration has finished")
  }

  void migrate(long structureId) {
    try {
      migrate0(structureId)
    } catch (Exception e) {
      log.error("Failed to migrate transforms for structure id=${structureId}.\n$
{trimStackTrace(e)}")
    }
  }

  void migrate0(long structureId) {
    def transformsJson = sps.getString(structureId, 'quick-transforms', '[]')
    def jsonSlurper = new JsonSlurper()
    def transforms = jsonSlurper.parseText(transformsJson)
    def newTransforms = [];
    transforms.each { t ->
      if (t.key != 'group') {
        newTransforms.add(t)
      } else if (t.transforms != null) {
        log.warn("Ungrouping '${t?.quick?.name}' to ${t.transforms.size()} transforms
in structure id=${structureId}")
        newTransforms.addAll(t.transforms)
```

```
      }
    }
    sps.setValue(structureId, 'quick-transforms', JsonOutput.toJson(newTransforms))
  }

  String trimStackTrace(Exception e) {
    return asList(e.stackTrace).subList(0, 50).join('\n    ') + '\n    ...'
  }
}

new QuickTransformsMigration(log: log).run()
```

Need help or have questions? Contact Tempo Support[289].

7.8 Structure 7.1 Release Notes

22th of July, 2021

Structure 7.1 adds support for comments in formulas and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[290] (no installation required)

7.8.1 Version Highlights

Structure 7.1 adds the following capabilities:

- Formulas(see page 228) now support comments[291] from issues and worklog's (including Tempo Timesheets)
- Notes column[292] and Quick transformation[293] data can now be included when copying a structure
- Security patch

7.8.2 Changes in Detail

7.8.2.1 Additional Updates

- Fixed: Epic SumUp[294] compatibility
- Fixed: Power Custom Fields[295] for Jira compatibility

289 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
290 https://demo-structure.almworks.com/
291 https://wiki.almworks.com/display/structure/Sample+Formulas#SampleFormulas-Comments
292 https://wiki.almworks.com/pages/viewpage.action?pageId=32228071
293 https://wiki.almworks.com/pages/viewpage.action?pageId=32223328
294 https://marketplace.atlassian.com/apps/1213091/epic-sum-up-simple-project-management?hosting=server&tab=overview
295 https://marketplace.atlassian.com/apps/1210749/power-custom-fields-for-jira

- Fixed: Advanced Roadmaps 3.29.7 compatibility
- Fixed: generators did not catch changes in some JQL functions (for example, openSprints(), releasedVersions(), etc.)

7.8.3 Supported Versions

Structure 7.1 and all extensions **support Jira versions 8.5 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[296] 2.7+
- Structure.Pages[297] 1.6+
- Structure.Testy [298]2.6+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.8.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.8.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [299]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.8.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–7.0 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

296 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
297 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
298 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
299 https://wiki.almworks.com/display/structure2gmaster/Download

7.8.5 Enterprise Deployment Notes

7.8.5.1 Testing on Staging Environment

In terms of stability and performance, this release does not bring significant changes compared to version 7.0.

There are no particular special areas of interest for load testing and stress testing Structure 7.1. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[300].

7.8.6 API Changes in Structure 7.1

7.8.6.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
8.5+	17.10.0

See Structure API Versions[301] for full version information and downloads.

7.8.6.2 2. Compatible Changes in the Java API

We have added `CoreEffects.setSprint(Issue, Long)` method that creates a description of an effect for moving issue to a sprint with a provided id.

`EffectorProcessManager.validateStartProcess(String, Map<String, Object>, ForestSpec)` method has been added for validating a one-off effector process without starting it.

We have clarified the meaning of `Effector.isAvailable()` method, please refer to the Javadoc[302] for details.

`BulkAccessibleItemType` interface has been added to support bulk items loading. Bulk items loading is used to optimize item attributes loading so implementing this interface by an item type class might improve attributes loading performance.

`ItemResolver` has new methods for resolving multiple items, bulk resolve is used when it is supported by an item type:

- `resolveItems(Collection<ItemIdentity>, Class<T>, BiConsumer<ItemIdentity, T>)`
- `resolveItemsUnchecked(Collection<ItemIdentity>, BiConsumer<ItemIdentity, Object>)`

300 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
301 https://wiki.almworks.com/display/structure2gmaster/Structure+API+Versions
302 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/Effector.html

7.9 Structure 7.0 Release Notes

7th of June, 2021

Structure 7.0 introduces major enhancements to formulas and adds performance improvements

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[303] (no installation required)

7.9.1 Version Highlights

Structure 7.0 introduces Expr 2 - a completely new level of sophistication and possibilities within formulas.

7.9.2 Changes in Detail

7.9.2.1 Formula Enhancements (Expr 2.0)

We've made major enhancements to Expr, Structure's formula language, allowing users to pull data from more places than ever before, including:

- issue links
- work logs
- related epic or stories
- parent task or sub-tasks (regardless of how the enclosing structure is built)
- versions – as a collection of values, and their properties
- sprints – as a collection of values, and their properties
- other multi-valued fields – as a collection of values
- properties of other fields – like users' full names

It's also now possible to embed queries (JQL and Structured JQL(see page 428)) within formulas.

There are multiple improvements in the Expr language, which support these new capabilities:

- Arrays - Formulas now supports arrays, or lists of values, such as Version fields with multiple values.
- Properties - Get the value of a particular property of an item using the following notation: `object.property`
- Chained Function Calls - Apply a sequence of functions to a value by using `value.function()` notation.
- User Functions - Define locally-used functions within a formula.
- Enhanced IF Expression - A convenient alternative to the `IF()` function, helping make formulas more readable.
- Concatenation Operator - Join text strings together.
- Text Snippets - Generate text using variables and expressions.
- New System Functions - Many new functions have been introduced to work with the new features.

Documentation: Changes to Expr in Structure 7(see page 419), Formulas(see page 228)

303 https://demo-structure.almworks.com/

7.9.2.2 Additional Updates

- The new Group by Attribute(see page 175) generator replaces the old "Group by Text Attribute" generator. Now it supports formulas that return arrays and items.
- Extender performance improvements.
- Sprint field can be updated by Effectors.
- Fixed: Tree CustomField compatibility.

7.9.2.3 Important Considerations for Upgrade and Downgrade

Backward-incompatible changes in Expr

In making this new generation of the formula language, we had to make some backward-incompatible changes. Those changes are minor and they are very unlikely to break your formula, but it's possible.

If you have been using formulas before, please review the Backward Compatibility section of the Changes to Expr in Structure 7(see page 419).

Review S-JQL queries used in conjunction with formula-based Group by (Text) Attribute generators

After upgrade to Structure 7 some formulas may return array or item values instead of text. If such formula is used in a Group by Attribute generator, and the formula result is a user, a priority, a status or some other item, then instead of folders the generator will create group rows representing those items naturally – as users, priority values, status values. This may break some existing S-JQL queries if they use "`folder()`" constraints.

If you encounter this problem, the fix is to replace "`folder()`" constraint with the corresponding "`item()`" constraint.

Downgrading from Structure 7

If you use the new formula capabilities introduced with Expr 2, and then downgrade to Structure 6 or an earlier version, the new formulas will stop working. The result of the calculation would most likely be an error, but also might be an undefined value or an incorrect result in some rare cases.

For that reason, if you use the new formulas to configure generators, the downgrade may result in an incorrectly built structure. For example, if you use Group... menu to add grouping by Sprint Name, it would create a Group by Attribute generator, configured with the "`sprint.name`" formula, which will not work in the previous versions of Structure. To fix the problem, you will need to re-create the generator.

Should you decide to downgrade, please review the formulas added to your Jira while using Structure 7.

7.9.3 Supported Versions

Structure 7.0 and all extensions **support Jira versions 8.5 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[304] 2.7+
- Structure.Pages[305] 1.6+

304 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
305 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview

- Structure.Testy [306] 2.5+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.9.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.9.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [307] page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.9.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.9.5 Enterprise Deployment Notes

7.9.5.1 Extender Performance Improvements

We have improved the performance of Linked Issues, Stories under Epic and Sub-task extenders once again. This time the changes were made to the performance of incremental updates and to an invisible part of the Structure app's work - when Structure checks if changes should be applied to a structure or not. Actual numbers will depend on the structure configuration, amount of issues in Jira, amount of changed issue, database performance, etc. However, in our tests we saw a 15-20% speedup of incremental updates.

306 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
307 https://wiki.almworks.com/display/structure2gmaster/Download

7.9.5.2 Event processing in Data Center DATA CENTER

In the Data Center environment, Structure running on one node needs to let Structure running on other nodes know when an item (an issue or some other object) changes. This "change stream" is communicated to other nodes via the database, asynchronous caches and the occasional use of a global, one-per-cluster lock.

Normally, each change is written into the database immediately when it happens – in the "event listener". That code runs in the same execution thread as the change itself, typically as a response to a user's action. From time to time we received support requests where the Jira global locking subsystem failed and there were certain issues with the database. This made the writing to the change stream "hang", which, in turn, made user request threads hang, which led to Jira being unresponsive.

In Structure 5.1[308], an alternative implementation of this subsystem was introduced, which never blocks a user request thread. All global lock operations and writing to the database happen in a separate, dedicated thread of execution.

This feature was experimental and not enabled by default. We have fixed the only problem that was found since Structure 5.1 - *ConcurrentModificationException* that was happening in rare cases. In Structure 7.0 we made this feature **enabled by default**. There is still a dark feature that allows you to return to the synchronous event processing mechanism: set `structure.delegatingItemTracker.enableAsyncEventsQueue` to `false`. For more information, see Advanced Configuration and Dark Features(see page 974).

To test this change, you can make issue changes on one node while a user observes a structure with JQL-based automation on another node. In a few seconds, the user should see the most up-to-date information in Structure.

7.9.5.3 Fix for CDN based on Microsoft Azure

We recently worked on an incident where Jira was configured to use a CDN server based on Microsoft Azure. In such an environment, Structure breaks the loading of CSS resources on Structure Board, Issue pages, and some other pages. In Structure 7.0, this problem was fixed.

To test this fix, you can just open Structure Board and see that it looks as expected.

7.9.5.4 Expression Language Performance Aspects

Structure 7.0 introduces a new version of Expression Language that allows you to use Issue Links, Worklogs and other items and data types including arrays. It allows you to construct more complicated formulas with arrays, JQL filters and user functions. This can cause potential performance problems connected with the amount of data (array size) and formula execution steps count. Structure 7.0 applies bulk data load for links, worklogs and JQL. It also has limits for array size, call count and call depth in formulas. But it is still possible to create a formula complicated enough to be slow. So please keep this in mind.

7.9.5.5 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 7.0. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[309].

308 https://wiki.almworks.com/display/structure2gmaster/Structure+5.1+Release+Notes
309 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.9.6 API Changes in Structure 7.0

7.9.6.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
8.5+	17.9.0

See Structure API Versions[310] for full version information and downloads.

7.9.6.2 2. Compatible Changes in the Java API

Several new item types have been introduced in Structure 7.0. `CoreItemTypes` has corresponding constants for them: `ATTACHMENT`, `COMMENT`, `WORKLOG`, `ISSUE_LINK`, `ISSUE_LINK_TYPE`. Also there are new methods in `CoreIdentities` to create item identities of these types.

Attribute Grouper replaces Text Grouper in Structure 7.0. New `GROUPER_ATTRIBUTE` constant have been added to `CoreStructureGenerators` and `GROUPER_TEXT` has been marked `@Deprecated`.

`CoreIdentities` has new methods to create item identities using `String` id: `priority(String)`, `status(String)`, `resolution(String)`, `label(String)`, `issueType(String)`. Also several new methods have been added to `CoreIdentities` for checking item type of `ItemIdentity`.

We have renamed `confirm` parameter to `requiresConfirmation` in `EffectorProcessManager.startProcess()` methods and fixed a mistake in Javadoc describing this parameter.

Effector components are now available in `StructureComponents`:

- `StructureComponents.getEffectorInstanceManager()`
- `StructureComponents.getEffectorProcessManager()`

7.10 Structure 6.6 Release Notes

18th of March, 2021

Structure 6.6 adds Attribute Filter, performance improvements and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[311] (no installation required)

310 https://wiki.almworks.com/display/structure2gmaster/Structure+API+Versions
311 https://demo-structure.almworks.com/

7.10.1 Version Highlights

Structure 6.6 adds:

- Attribute filter
- Grouper performance improvement
- Security patch

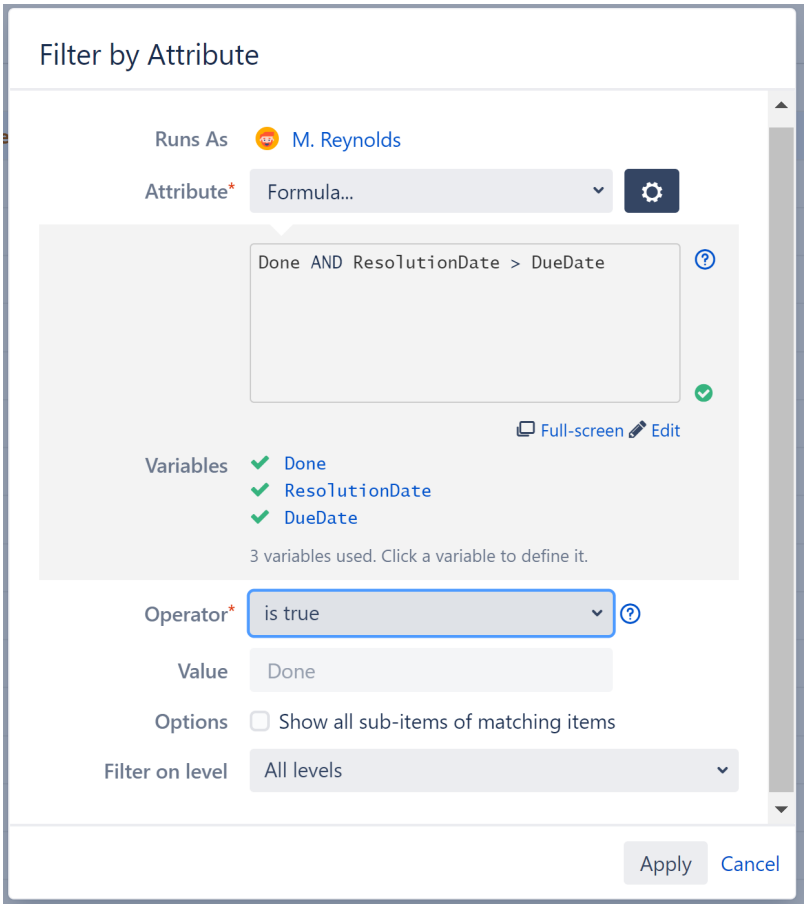
7.10.2 Changes in Detail

7.10.2.1 Attribute Filter

The Attribute filter allows you to limit the issues in a structure based on the values in a specific attribute. You can filter a structure based on:

- **Formulas** - Use a formula to compare Jira values, and filter the structure based on those results.
- **Time in Status** - Filter issues that are taking too long in a particular status.
- **Text filter** - Filter a formula or notes column for specific words or strings.
- **Gantt Attributes** - If you have Structure.Gantt[312] installed, you can also filter by Gantt attributes (such as the Gantt Start or Finish date, or issues in the Critical Path). *Note: Gantt attributes are only available when using the Filter by Attributes* Transformation[313]*; Automation does not support Gantt attributes.*

312 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale?hosting=server&tab=overview
313 https://wiki.almworks.com/display/structure2gmaster/Transformations



The Attribute filter can be used with Generators(see page 140) and Transformations(see page 215).

Documentation: Attribute Filter(see page 160)

7.10.2.2 Additional Updates

- Added formula autocomplete for version and sprint attributes.
- Fixed: Anonymous users received an error when trying to access a structure that was not available. They are now redirected to the Jira login page.
- Fixed: When a structure contains multiple groups containing the same issues, it was not possible to manually reorder just one group.

7.10.3 Supported Versions

Structure 6.6 and all extensions **support Jira versions 8.5 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[314] 2.7+

314 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

- Structure.Pages[315] 1.6+
- Structure.Testy [316] 2.5+

Cloud customers can learn more about our products on the “Cloud” tab of our marketplace listing.

7.10.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.10.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [317]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.10.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.5 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.10.4.3 Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.5, either from the Atlassian Marketplace or our Download[318] page.

---

315 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
316 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
317 https://wiki.almworks.com/display/structure2gmaster/Download
318 https://wiki.almworks.com/display/structure2gmaster/Download

3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.10.5 Enterprise Deployment Notes

7.10.5.1 Grouper Performance Improvements

We have improved the performance of most Group generators that group issues on levels deeper than level 1. Actual numbers will depend on the structure configuration and amount of issues on the selected grouping levels; however, in our tests, a structure with 10K issues on the grouping levels, using an Agile Epic grouper and Field grouper, ran 2 times faster in Structure 6.6 compared to version 6.5.2.

You can use the Performance Audit Log to find a generator's execution time.

7.10.5.2 Performance Audit Log fix

This version also delivers a fix for the Performance Audit Log (PAL) bug for OpenJDK version 9 and later. PAL failed if your had a structure with a JQL filter based on another JQL filter. The Performance Audit Log is important for Structure performance monitoring, optimization, and problem detection.

7.10.5.3 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 6.6. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[319].

7.10.6 API Changes in Structure 6.6

7.10.6.1 1. Minor Java API Release

There is one API addition coming with this release. The change is backwards-compatible, so any code using API 17.x should work fine.

319 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Jira Version	New API Version
8.5+	17.8.0

See Structure API Versions[320] for full version information and downloads.

7.10.6.2 2. Compatible Changes in the Java API

We have added a module key constant for Attribute filter: `CoreStructureGenerators.FILTER_ATTRIBUTE`

7.11 Structure 6.5 Release Notes

> **1st of February, 2021**
>
> Structure 6.5 adds Perspective link sharing, Advanced Roadmaps Team Effector and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[321] (no installation required)

7.11.1 Version Highlights

Structure 6.5 adds:

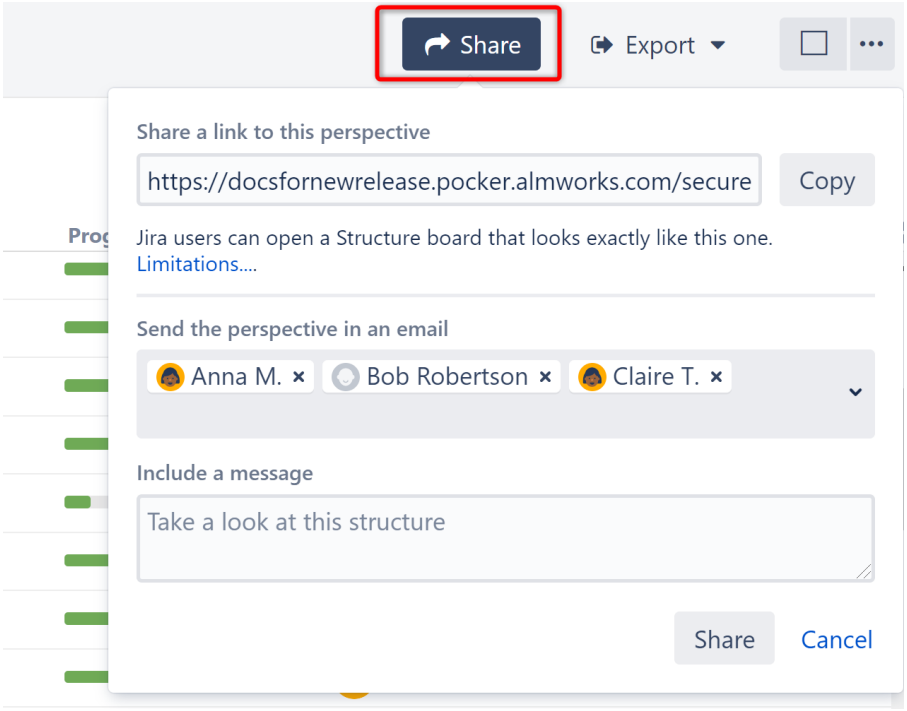
- Perspective sharing via email
- Advanced Roadmaps (Portfolio) Team Effector
- Additional improvements and fixes

7.11.2 Changes in Detail

7.11.2.1 Perspective sharing via email

Let other users view structures with the same information you see, by sending perspective links directly to their email.

320 https://wiki.almworks.com/display/structure2gmaster/Structure+API+Versions
321 https://demo-structure.almworks.com/



Documentation: Sharing a Perspective(see page 97)

7.11.2.2 Advanced Roadmaps (Portfolio) Team Effector

It is now possible to update the Team field from Advanced Roadmaps using the Attribute to Issue Field Effector(see page 179).

7.11.2.3 Additional Updates

- When exporting a structure(see page 558) with horizontal scrolling (see page 503)enabled, the pinned column order can now be maintained with the export.
- The Sort generator icon now indications whether the sort is in ascending or descending order.
- Fixed: Could not group by Tempo 11 attributes using the Text Attribute grouper(see page 732).
- Fixed: Text color for disabled options in Structure global permissions was too light to read.
- Fixed: Dates in the Work Logged column displayed incorrectly when a user's time zone was different from the system time zone.
- Fixed: Loop markers created by synchronizers can be stored in DB after deleting the tasks they are pointed at.
- Additional performance improvements.

7.11.3 Supported Versions

Structure 6.5 and all extensions **support Jira versions 8.5 or later**. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[322] 2.7+
- Structure.Pages[323] 1.6+
- Structure.Testy [324] 2.5+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.11.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.11.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [325]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.11.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.11.4.3 Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.

---

322 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
323 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
324 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
325 https://wiki.almworks.com/display/structure2gmaster/Download

2. Download and install Structure 6.5, either from the Atlassian Marketplace or our Download[326] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.11.5 Enterprise Deployment Notes

7.11.5.1 Additional Attributes subsystem permission checks

We have added additional permission checks for attribute calculation in structures that use substructures. If you have such structures with aggregate attributes (columns), please check to see that attribute values still load quickly.

7.11.5.2 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 6.5. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[327].

7.11.6 API Changes in Structure 6.5

7.11.6.1 1. Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
8.5+	17.7.0

See Structure API Versions[328] for full version information and downloads.

7.11.6.2 2. Compatible Changes in the Java API

We have added project category items support. Items can be used in attribute trails or generator `ItemChangeFilter` to detect project category changes.

326 https://wiki.almworks.com/display/structure2gmaster/Download
327 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
328 https://wiki.almworks.com/display/structure2gmaster/Structure+API+Versions

Corresponding type constant:

- `CoreItemTypes.PROJECT_CATEGORY`

Methods to create project category items:

- `CoreIdentities.projectCategory(long)`
- `CoreIdentities.projectCategory(ProjectCategory)`

7.11.7 Structure 6.5.1 Release Notes

8th of February, 2021

Structure 6.5.1 adds Jira 8.15 compatibility

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[329] (no installation required)

7.11.7.1 Version Highlights

Structure 6.5.1 is a patch version based on Structure 6.5. It provides:

- Jira 8.15 compatibility
- Fixed: Confluence gadget is unavailable for anonymous users
- Fixed: Excessive memory consumption in specific cases of Structure REST API usage.

7.11.7.2 Supported Versions

Structure 6.5.1 and all extensions support Jira versions 8.5 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[330] 2.7+
- Structure.Pages[331] 1.6+
- Structure.Testy [332] 2.5+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

329 https://demo-structure.almworks.com/
330 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
331 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
332 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.11.7.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [333]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.5 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.5.1, either from the Atlassian Marketplace or our Download[334] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

333 https://wiki.almworks.com/display/structure2gmaster/Download
334 https://wiki.almworks.com/display/structure2gmaster/Download

7.11.7.4 Enterprise Deployment Notes

We have fixed a memory leak that caused an OutOfMemoryError in some specific cases of Structure REST API usage. If you use Forest Resource(see page 1105) to move rows in the forest, and if you do thousands of such move operations, please test your scenarios . You should see significantly less memory consumption. Please note that this issue was related to move actions only. See Forest Resource(see page 1105) for more details.

Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 6.5.1. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[335].

7.11.8 Structure 6.5.2 Release Notes

23rd of February, 2021

Structure 6.5.2 is a patch release for 6.5. It fixes the full screen formula editor and more.

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[336] (no installation required)

7.11.8.1 Version Highlights

Structure 6.5.2 is a patch release based on Structure 6.5. It provides:

- Fixed: Full Screen mode is unavailable when editing formulas for generators and transformations
- Fixed: Error is displayed when using Structure.Testy attributes within a column
- Fixed: UI problem when editing attribute columns in the Confluence gadget

Known issue: Structure Gadget is not supported in IE11

7.11.8.2 Supported Versions

Structure 6.5.2 and all extensions support Jira versions 8.5 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[337] 2.7+
- Structure.Pages[338] 1.6+

335 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
336 https://demo-structure.almworks.com/
337 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
338 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview

- Structure.Testy [339] 2.5+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.11.8.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [340] page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.5.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.5.1, either from the Atlassian Marketplace or our Download [341] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).

339 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
340 https://wiki.almworks.com/display/structure2gmaster/Download
341 https://wiki.almworks.com/display/structure2gmaster/Download

4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.12 Structure 6.4 Release Notes

> **7th of December, 2020**
>
> Structure 6.4 adds Fix Version and Sprint data in the Summary column and formulas, performance improvements and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[342] (no installation required)

7.12.1 Version Highlights

Structure 6.4 adds:

- Fix version and sprint details can now be shown in the Summary column and used in formulas
- Option to show names, avatars or both in user columns
- Additional improvements and fixes

7.12.2 Changes in Detail

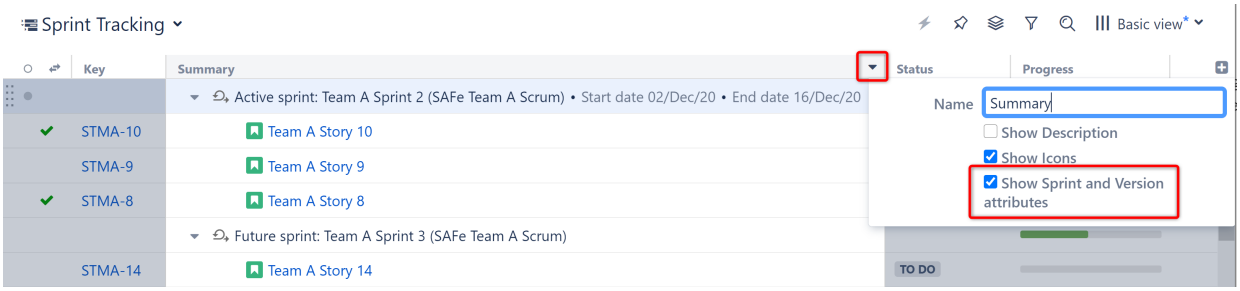
7.12.2.1 Fix Version and Sprint data

The following attributes can now be shown in the Summary column, or used in formulas and effectors:

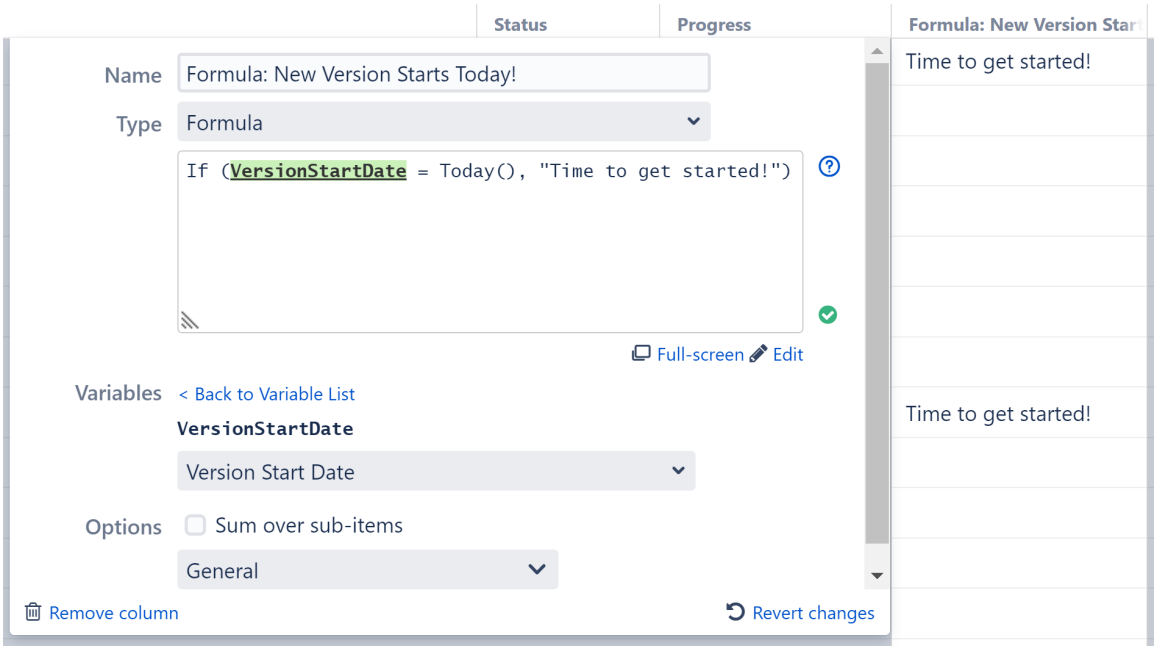
- Fix version start date
- Fix version release date
- Fix version description
- Sprint start date
- Sprint end date
- Sprint Goal

To show these attributes in the summary column, click the arrow icon in the column header and check **Show Sprint and Version attributes**.

342 https://demo-structure.almworks.com/



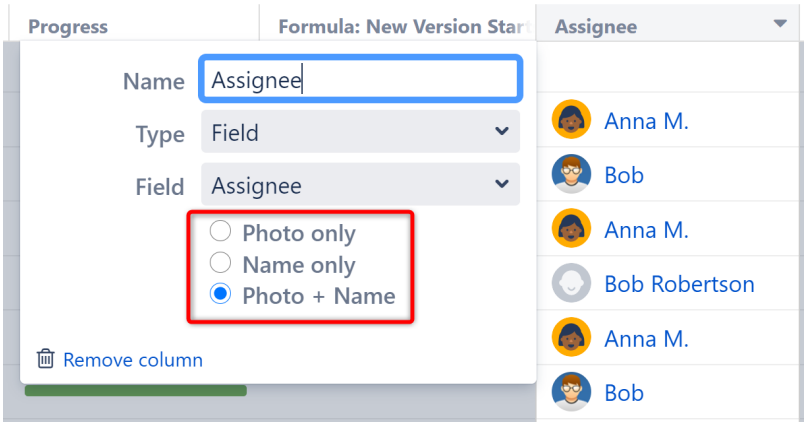
To include these in a formula, map a variable to the appropriate version or sprint attribute.



> ⚠ To view fix version or sprint data (or use these attributes in a formula), issues must be grouped by fix version or sprint. The version/sprint attributes are only available for version/sprint rows.

7.12.2.2 Photo and Name in User Fields

It is now possible to choose whether user fields (Assignee, Creator, Reporter, custom user-picker fields) show the user's name, photo (avatar) or both.



7.12.2.3 Additional Updates

- It is now possible to select multiple projects and version when using the Backlog Structure wizard.
- Extender performance improvements. Learn more(see page 744)
- Synchronizers now check the structure size before creating new rows. Learn more(see page 744)
- Fixed: When working on the Project Page, Structure did not auto-switch to the default.
- Fixed: When viewing a structure in two panels, changing an issue's attributes in one panel would cause attributes in the other to be marked as outdated (grayed out).

7.12.3 Supported Versions

Structure 6.4 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Note, that 6.4 is the last version that supports Jira 7.13; the next major version will require Jira 8.5+.

Compatible plugins versions:

- Structure.Gantt[343] 2.7+
- Structure.Pages[344] 1.5+
- Structure.Testy [345] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.12.4 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

343 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
344 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
345 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.12.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [346]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.12.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.3 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.12.4.3 Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.4, either from the Atlassian Marketplace or our Download[347] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.12.5 Enterprise Deployment Notes

Structure 6.4 introduces a few changes and improvements especially important for large-scale Jira Server and Jira Data Center instances.

346 https://wiki.almworks.com/display/structure2gmaster/Download
347 https://wiki.almworks.com/display/structure2gmaster/Download

7.12.5.1 Extender Performance Improvements

We have updated our structure generation algorithm, which now allows extenders to preload the data for all rows on the current level of the structure at once, even when those rows belong to different sub-trees. Combined with new ways of loading data for multiple issues at once, the new algorithm has led to significant performance improvements for extender-based structures.

All four extenders that ship with Structure have been updated to take advantage of the new generation algorithm, and in our internal tests we have seen them work several times faster in certain scenarios. If you have large, multi-level structures built with extenders, we advise that you verify their performance on a staging system before upgrading.

7.12.5.2 Confluence Gadget Updates

Continuing the work started in Structure 6.3, we have made a few adjustments in the Structure gadget when it's embedded in Confluence pages. On Jira Data Center, we have increased the gadget's poll interval to 30 seconds and also increased the window size for loading attribute values. With fewer REST requests from the gadget to the server, the probability of reaching a different cluster node should now be reduced. Also, we have made various aspects of the gadget's communication with the server configurable with dark features.

7.12.5.3 Other Improvements and Fixes

We have identified and fixed a bug where a simple sorter by an issue field would in certain cases cause a full re-generation of the structure on each request, making it constantly busy and, in certain cases, unusable.

Another problem we have fixed in this version was related to synchronizers. Since Structure 4.1, there is a hard limit on the size of static structures (100,000 rows by default). Synchronizers are also not allowed to exceed the limit, but we have discovered that they would still insert records for new rows into the database, even when it wouldn't be possible to add those rows to the structure. Since synchronizers run quietly in the background, this could cause a lot of garbage to accumulate in the database. We have fixed the problem, but if you have synchronizers operating on structures that have reached the limit, we advise you to check the size of the `AO_8BAD1B_ROW` table in your database.

7.12.5.4 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 6.4. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[348].

348 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.12.6 API Changes in Structure 6.4

7.12.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.6.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.12.6.2 Compatible Changes in the Java API

We have added two new constants to support the "creator" attribute:

- `CoreAttributeSpecs.CREATOR` is the attribute spec that returns the creator of an item as an `ApplicationUser`;
- `CoreAttributeSpecs.Id.CREATOR` is the attribute ID for the "creator" attribute.

7.13 Structure 6.3 Release Notes

8th of October, 2020

Structure 6.3 adds the Status Rollup Effector, default views, Project Category grouper and more.

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[349] (no installation required)

7.13.1 Version Highlights

Structure 6.3 adds:

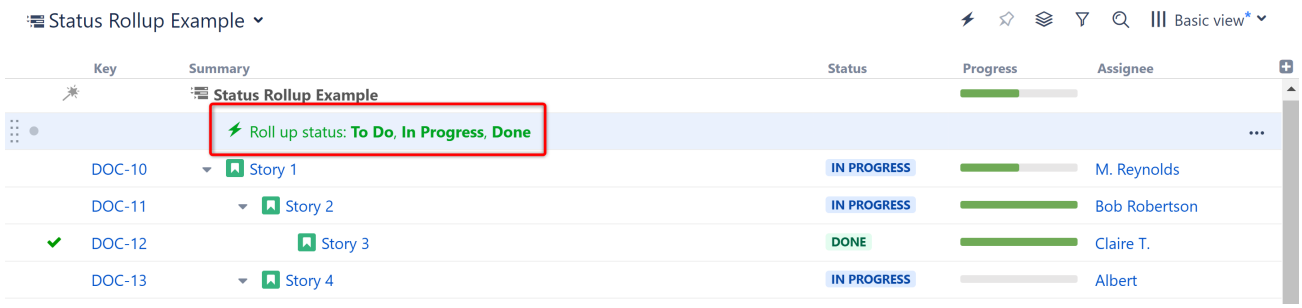
- Status Rollup Effector
- Default view for structures
- Group by Project Category
- Improved Information Center
- Ability to edit numeric custom fields through Structure Gadget
- Additional improvements

349 https://demo-structure.almworks.com/

7.13.2 Changes in Detail

7.13.2.1 Status Rollup Effector

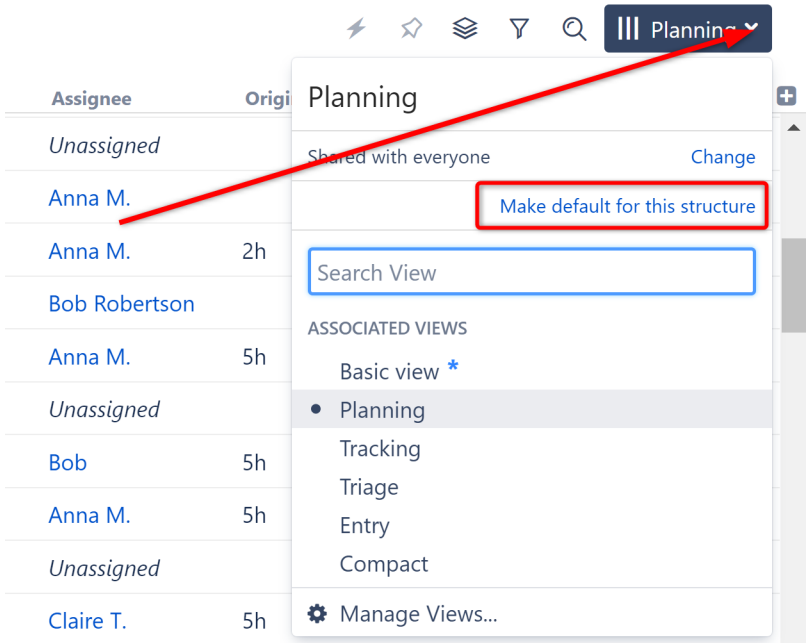
This new effector allows you to automatically update parent issue statuses based on the earliest status of their sub-issues.



Documentation: Status Rollup Effector(see page 182)

7.13.2.2 Default Views

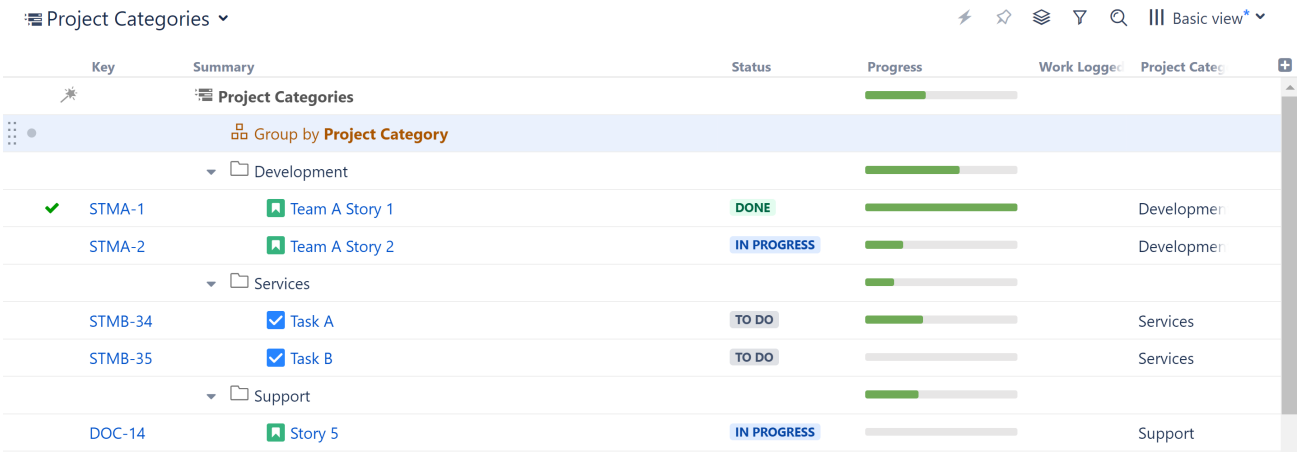
It is now possible to select a default view for each structure right from the Views menu. When set, users who open the structure for the first time will see the default view. Additionally, if there is no recently-selected view available for the structure, it will open with the default view.



Documentation: Views Menu(see page 507)

7.13.2.3 Group by Project Category and ScriptRunner Single Issue Picker

It is now possible to group items by Project Category and the ScriptRunner Single Issue Picker field, using Group Automation.



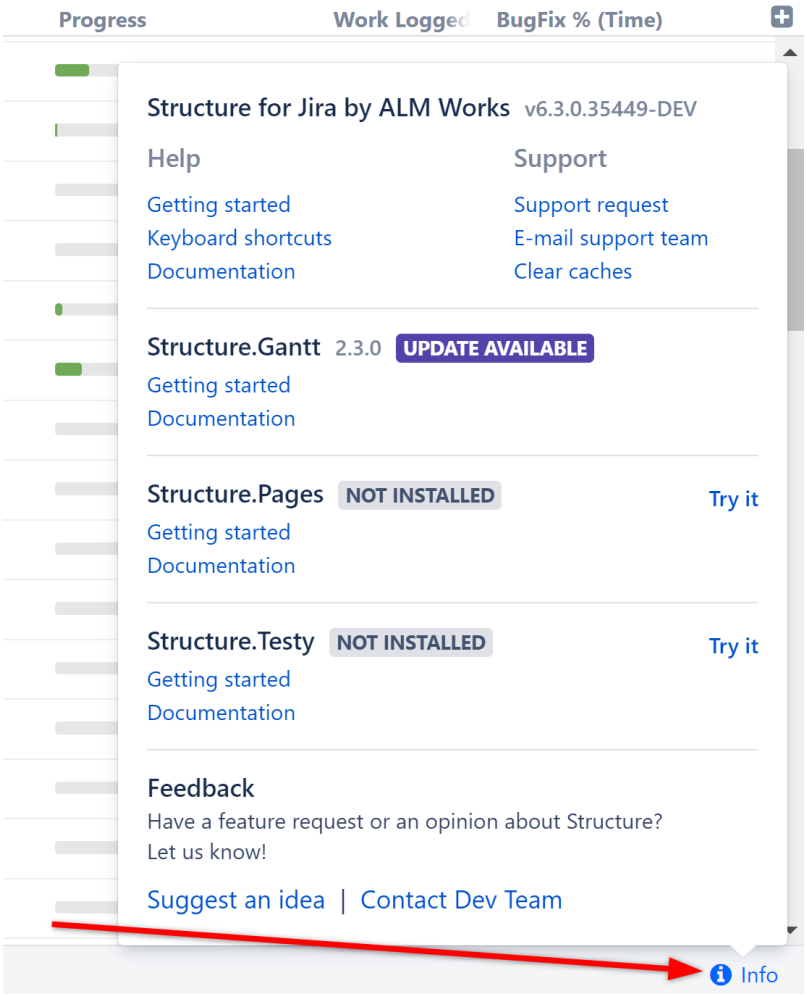
To Group by Project Category, select **Automation | Group by Text Attribute**, and then select **Group by Project Category**. If you have ScriptRunner installed and have defined Single Issue Pickers, they will appear under the **Automation | Group by...** menu.

- Project Category and the ScriptRunner Picker field can also be added as Structure columns and used in formulas.
- ScriptRunner Picker values can be updated within the structure by dragging an issue to another group or using inline editing (when added as a column).

Documentation: Group Generators(see page 169), ScriptRunner(see page 577)

7.13.2.4 Improved Information Center

The improved Info window now includes additional links and information about Structure extensions.



7.13.2.5 Additional Updates

- Structure now remembers the last view used with each structure, and returns to that view the next time the structure is opened.
- A new hotkey ('vvv') returns to the previous view.
- It is now impossible to edit gadget in Confluence.
- It is now possible to update numeric fields in the Dashboard gadget.
- It is now possible to include the backlog when using the Sprint Filter Transformation.
- Query Match is now a resizable field in Structure formulas.
- Attribute system now reads the values of numeric, date, and date-time custom fields from the Lucene index. Learn more.(see page 750)
- Fixed: Data was lost when saving changes to a Memo.
- Fixed: Confluence gadget displayed values unrelated to the displayed structure. Learn more.(see page 750)
- Fixed: Export failed due to Lock Timeout on busy systems. Learn more.(see page 751)

7.13.3 Supported Versions

Structure 6.3 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Desk).

Compatible plugins versions:

- Structure.Gantt[350] 2.4+
- Structure.Pages[351] 1.5+
- Structure.Testy [352] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.13.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.13.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [353]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.13.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.13.4.3 Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

---

350 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
351 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
352 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
353 https://wiki.almworks.com/display/structure2gmaster/Download

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[354] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.13.5 Enterprise Deployment Notes

Structure 6.3 introduces a few changes and improvements especially important for large-scale Jira Server and Jira Data Center instances.

7.13.5.1 Reading Custom Fields from the Index

Starting with Structure 6.3, the attribute system reads the values of numeric, date, and date-time custom fields from the Lucene index instead of the database. Reading a single field for many issues from the index is usually much faster than loading a lot of issue objects one by one from the database. Structure generators have been using this technique for a long time, and now we're bringing it to the attribute subsystem.

This improvement is especially important for aggregate values like sums, where Structure sometimes needs to load the values for many more issues than the user sees on the screen to calculate the sum. If you have big structures with thousands of issues and you calculate aggregates based on numeric, date and date-time custom fields on those structures, we advise that you verify the performance of those aggregate columns in a staging environment before upgrading.

7.13.5.2 Cluster Node Tracking in Structure Gadget in Confluence

The Structure gadget, unlike most other Jira gadgets, is a live web application, and it uses multiple REST requests to load the data and keep it up-to-date. When a Structure gadget is inside a Confluence page, those REST requests have to be proxied by Confluence: the gadget calls Confluence, which then calls Jira and forwards its reply back to the gadget. For each forwarded request, Confluence authenticates the current user, creating a new, single-use user session. In a Jira Data Center instance, the load balancer can route these requests to different cluster nodes, so it is possible, for example, that the gadget loads the structure from one node, but the values shown in the grid come from a different node.

Structure was not designed to operate in these conditions. This problem was hard to notice in earlier versions of Structure, but since Structure 6.0, completely unrelated values may be shown in the grid, giving the illusion that a different structure is displayed.

In Structure 6.3, we've updated the gadget to keep track of the cluster node: it remembers the node from which it received its first reply, and if later a reply comes from a different node, the gadget will ignore it and retry the request, hoping that the load balancer will redirect it to the original node. If the gadget is unable to reach the original node after multiple attempts, it will stop updating the data and display an error message. If the original node is down, reloading the page will reinitialize the gadget and "bind" it to a different node.

Also, the gadget will always switch to read-only mode if it's on a Confluence page and the Jira instance is a Data Center instance. This solution is only a temporary workaround. We are still researching better ways of integrating Structure into Confluence pages.

---

354 https://wiki.almworks.com/display/structure2gmaster/Download

7.13.5.3 More Reliable Export Under Load

We have seen a few cases where, on a busy system, exporting a structure to an Excel file or printable HTML would fail due to a lock timeout because the exporting thread would not be able to lock and refresh the structure for 30 seconds. Users would then have to retry or, in the most extreme cases, schedule their exports during a quiet period.

In Structure 6.3, we've made the export process more robust: it will now wait for at least 10 minutes to let the other threads finish and get access to the structure. Also, after 2 minutes of waiting time, the exporter will try to switch to a "fast path": if there's already a recent version of this structure in the cache, it will be exported without waiting for the lock (accompanied by a warning that the exported version may be outdated). We hope that these changes improve users' experience when exporting large, complex structures under heavy load.

7.13.5.4 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 6.3. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[355].

7.13.6 API Changes in Structure 6.3

7.13.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.4.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.13.6.2 Compatible Changes in the Java API

The `MapObject` class has received new versions of `getLong()`, `getInt()`, and `getBoolean()` that accept a default value. These are added for convenience.

`StructureViewManager.makeDefaultForStructure()` is a new method that will associate a view with a structure and make it the default view for the given page, all in a single call.

355 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.13.7 API Changes in Structure 6.3.2

7.13.7.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.5.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.13.7.2 Compatible Changes in the Java API

The `StructurePluginHelper` interface has two new methods: `isSystemAdmin()` and `isSystemAdmin(ApplicationUser)` check whether a user is a Jira system administrator. They complement the existing `isAdmin()` methods.

The `BackupOperation` interface has several new methods that allow you to do partial backups and back up to `OutputStream` instead of a file on the server:

- `setBackupStructureIds(LongIterable)`
- `setBackupViewIds(LongIterable)`
- `setBackupAppConfiguration(boolean)`
- `setBackupPerspectives(boolean)`
- `setBackupFavorites(boolean)`
- `backupToStream(OutputStream)`

Three new methods have been added to the `CoreIdentities` class to support memos and improve support for folders:

- `memo(long)`
- `isMemo(ItemIdentity)`
- `isFolder(ItemIdentity)`

7.13.8 Structure 6.3.1 Release Notes

10th of October, 2020

Structure 6.3.1 is a patch release for 6.3. It fixes an issue with mentioning multiple users on the Jira issue page.

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[356] (no installation required)

356 https://demo-structure.almworks.com/

7.13.8.1 Patch Release

Structure 6.3.1 is a patch release for Structure 6.3. It fixes an issue that prevented some users from including multiple user mentions within comments on the Jira issue page.

Upgrade from Structure 6.3 is highly recommended.

7.13.8.2 Supported Versions

Structure 6.3 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Desk).

Compatible plugins versions:

- Structure.Gantt[357] 2.4+
- Structure.Pages[358] 1.5+
- Structure.Testy [359] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.13.8.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [360]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.

---

357 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
358 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
359 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
360 https://wiki.almworks.com/display/structure2gmaster/Download

4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[361] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.13.9 Structure 6.3.2 Release Notes

18th of November, 2020

Structure 6.3.2 is a patch release for 6.3. It adds Jira Cloud Migration Assistant support and fixes a problem with custom fields.

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[362] (no installation required)

7.13.9.1 Patch Release

Structure 6.3.2 is a patch release for Structure 6.3. It provides:

- Migration assistance to migrate structures from Server/Data Center to Jira Cloud
- Fix to an issue that prevented Structure from reading the values of some custom fields

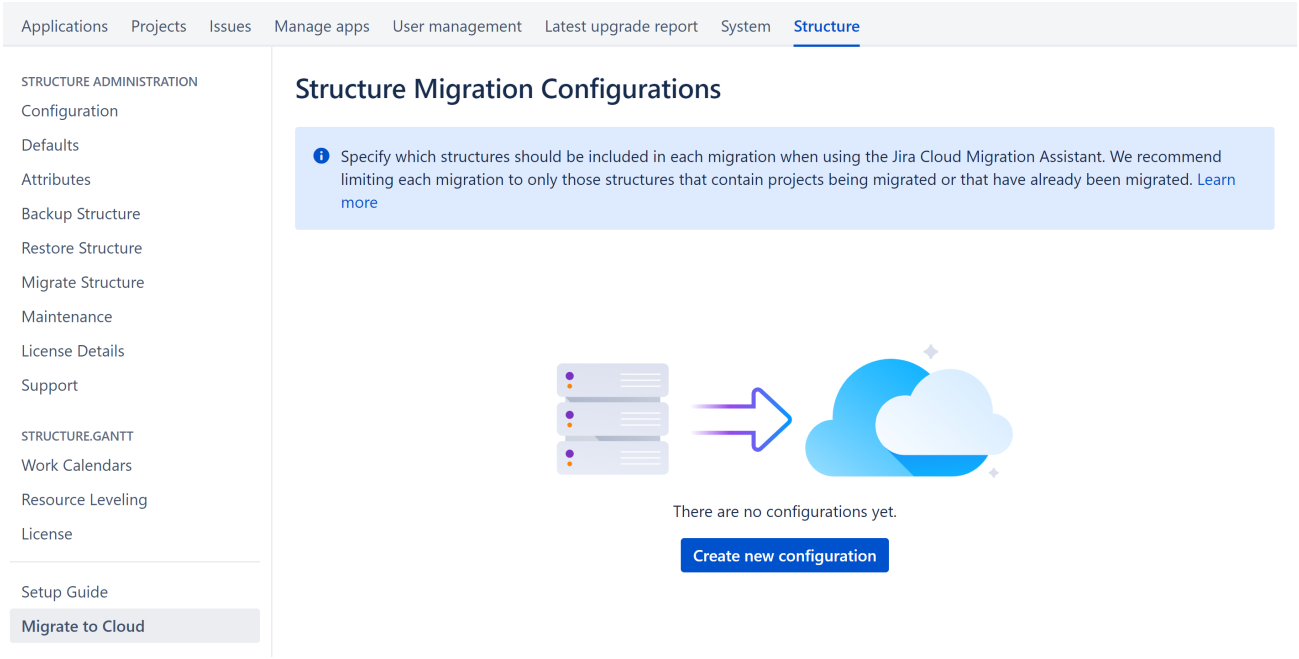
Upgrade from Structure 6.3 is highly recommended.

7.13.9.2 Changes in Detail

Migration Assistance

It is now possible to add Structure data to a migration when using the Jira Cloud Migration Assistant (JCMA).

361 https://wiki.almworks.com/display/structure2gmaster/Download
362 https://demo-structure.almworks.com/



Structure Migration Configurations allow you to associate specific structures with JCMA migrations. When you run JCMA, those structures will be recreated in Structure Cloud, based on:

- Manually added issues (see page 102)and folders(see page 104)
- Items added and organized through Generators(see page 140)
- The default view(see page 541) and associated views(see page 516) for each structure

Please note the following restrictions:

- Structure can only add issues from projects that have been migrated to cloud using the JCMA.
- At this time, some features are not available in Structure Cloud, including:
  - Certain types of Automation, including Effectors
  - Memos – any existing memos will be converted to folders in Structure Cloud

  For more information see: Comparison to Structure for Jira Server, Data Center and Service Management[363].

The Structure Cloud portion of the migration tool is still under development, and will be released shortly. At this time, it is only possible to prepare migration sets. As soon as the Structure Cloud update is released, any sets you've created can then be used to migrate to Structure Cloud.

Documentation: Migrate to Cloud(see page 930)

Additional Updates

- Fixed: Structure was unable to read the values of some custom fields. See Enterprise Deployment Notes(see page 757).

363 https://wiki.almworks.com/display/strcloud/Comparison+to+Structure+for+Jira+Server%2C+Data+Center+and+Service+Management

7.13.9.3 Supported Versions

Structure 6.3 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[364] 2.4+
- Structure.Pages[365] 1.5+
- Structure.Testy [366] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.13.9.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [367]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.3.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

---

364 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
365 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
366 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
367 https://wiki.almworks.com/display/structure2gmaster/Download

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[368] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.13.9.5 Enterprise Deployment Notes

Structure 6.3 introduced an optimization in the attribute system, which used the Lucene index to quickly read the values of numeric, date, and date-time custom fields. We have since seen a few support cases where the optimized code didn't work as expected, which resulted in Structure being unable to read the values of certain custom fields. We are working on resolving the problem, but in the meantime Structure 6.3.2 disables this optimization by default, and reads all custom field values from the database. If needed, the optimization can be enabled by setting a dark features(see page 974).

Otherwise, there are no particular special areas of interest for load testing and stress testing Structure 6.3.2. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[369].

7.14 Structure 6.2 Release Notes

10th of August, 2020

Structure 6.2 adds Effectors support for duration and multi-value fields, full-screen formula editor, migration improvements, last comment column and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[370] (no installation required)

7.14.1 Version Highlights

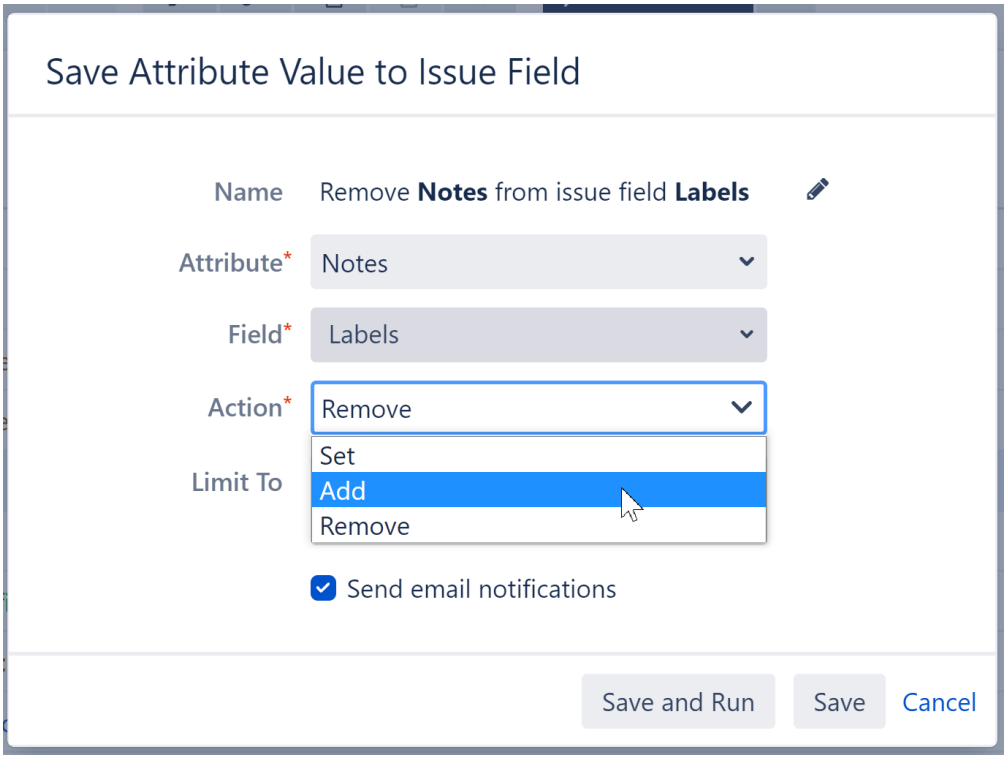
- Effectors can write values to duration and multi-value fields
- Full-screen formula editor
- Last Comment column

368 https://wiki.almworks.com/display/structure2gmaster/Download
369 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
370 https://demo-structure.almworks.com/

7.14.2 Changes in Detail

7.14.2.1 Effectors Can Write Values to Duration and Multi-Value Fields

Effectors can now write values to duration fields and multi-value fields. With multi-value fields, effectors can replace, add or remove values based on a selected attribute.



Documentation: Attribute to Issue Field Effector(see page 179)

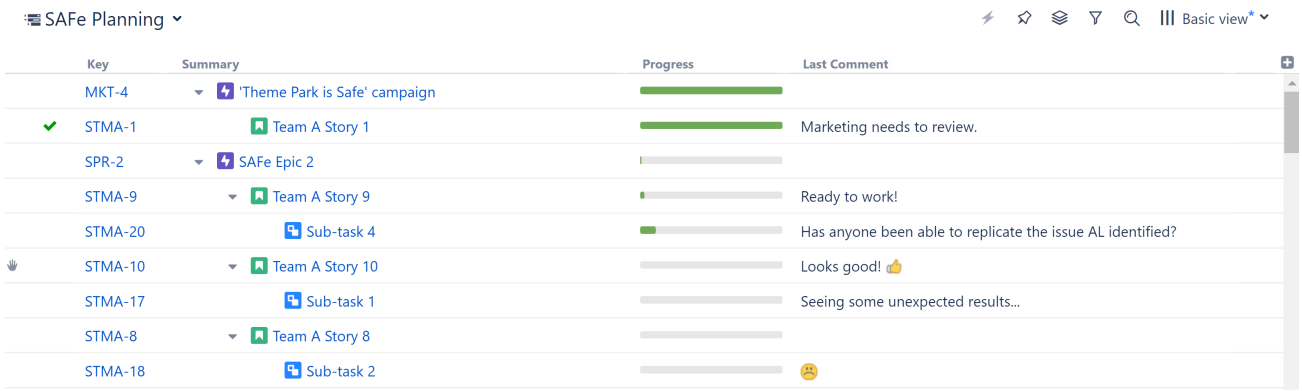
7.14.2.2 Full-Screen Formula Editor

The formula editor can be expanded to more easily write/review a formula.

Documentation: Creating a Formula(see page 229)

7.14.2.3 Last Comment Column

The Last Comment column displays the last comment left for each issue.



Documentation: Last Comment Column(see page 469)

7.14.2.4 Additional Updates

- Sort by Status (Automation and Transformation) now sorts by Status Category by default.
- On the Project Page, Structure now opens in the Single Grid layout by default.
- View modifications made with Grid + Details layout are now preserved when changing to Single Grid layout (on Structure Board and Project Page).
- When a client opens a structure for the first time, Structure now checks for recent cached versions on the server to reduce load times - see Enterprise Deployment Notes(see page 761) for details.
- Improved migration - see Enterprise Deployment Notes(see page 761) for details.
- Ability to enable users to manage permissions for groups they are not members of - see Advanced Configuration and Dark Features(see page 974)
- Fixed: Data was lost when saving changes to a Memo

7.14.3 Supported Versions

Structure 6.2 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Desk).

Compatible plugins versions:

- Structure.Gantt[371] 2.4+
- Structure.Pages[372] 1.5+
- Structure.Testy [373] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

371 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
372 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
373 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.14.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.14.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [374]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.14.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.14.4.3 Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[375] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.

374 https://wiki.almworks.com/display/structure2gmaster/Download
375 https://wiki.almworks.com/display/structure2gmaster/Download

5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.14.5 Enterprise Deployment Notes

Structure 6.2 introduces several improvements especially important for large-scale Jira Server and Jira Data Center instances.

7.14.5.1 Faster Initial Loading of Structures from the Cache

One of the changes in Structure 6.2 aims to improve responsiveness in certain scenarios. Previously, when a user opened a structure for the first time, the client would send a regular update request to the server to check if anything changed there since the last time this structure was generated and regenerate it as necessary. On large, busy systems and with complex structures that take a long time to generate, this would sometimes cause a significant delay before the user could see the structure.

Starting with Structure 6.2, when the client loads a structure from the server for the first time, it will send a special lightweight request that will first check if the server already has a recent version of this structure in its cache. If such a version exists, the server would quickly return it to the client, without checking for changes or waiting for ongoing regeneration to complete. After receiving and displaying this cached version, the client will then send a regular update request to load the most recent version of the structure in the background. This way, popular and complex structures should now load faster, especially on busy Jira instances with a high rate of change.

7.14.5.2 Migration Improvements

Some of our larger customers have used Migrate Structure to consolidate Structure data from several Jira instances that they were migrating into a single Jira Data Center instance. During the process, they encountered several problems that required manual intervention or writing ad-hoc scripts:

- Owners of Structure views were not migrated.
- Permission rules based on project roles were not migrated.
- "Apply Permissions" rules for structures were not migrated.

To address this, we've made the following improvements in Structure 6.2:

- View owners are now migrated.
- Role-based permission rules are now migrated.
- "Apply Permissions" rules are migrated if the structures they refer to are also migrated.
- Structure and view owners, as well as users mentioned in structure permission rules, are resolved by username first, to better deal with synthetic user keys in the latest Jira versions.
- When a permission rule cannot be migrated, Structure logs a warning and skips only this rule. Previously, all permission rules were skipped when any of them couldn't be migrated.

Please note that if you want all of these improvements, you need to migrate from backup files made with Structure 6.2, as they contain all the necessary information, such as usernames and project role names. Those improvements that don't need new information are also available when migrating from backup files made with earlier versions of Structure.

7.14.5.3 Testing on Staging Environment

To check faster initial loading of structures from the cache, you can try the following:

- Create a complex structure that takes a significant time to generate and update, for example, using a JQL inserter and several extenders. If you already have such a structure, open it and wait for it to load, so that it's generated and stored in the cache.
- Simulate a high rate of change, e.g. by making random changes to issues in that structure using a script.
- Try opening this structure in multiple browser tabs, using different user accounts. Notice the time it takes the structure to appear on the screen.

Apart from this change in the client-server communication, there are no particular special areas of interest for load testing and stress testing Structure 6.2. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[376].

7.14.6 API Changes in Structure 6.2

7.14.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.3.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.14.6.2 Compatible Changes in the Java API

`CoreEffects` has many new methods that generate effect descriptions for the new effect providers in Structure 6.2, which can update multi-valued fields, both built-in and custom. There are 30 new methods in total, please refer to the Javadoc[377] for details.

There are two new constants in `CoreEffectorParameters`:

- `SEND_NOTIFICATIONS` controls whether e-mail notifications are sent when effects are applied.
- `COLLECTION_OPERATION` controls whether an effector should add, remove, or replace values when working with a multi-valued field.

The Attribute to Issue Field effector supports both parameters.

`StructureConfiguration` has new setter methods that accept project IDs and permission subjects where previously it accepted only string-encoded representations thereof:

- `setPickedProjectIds(LongIterable)`
- `setPickedProjectIds(Collection<Long>)`
- `setPermissionSubjects(StructureAppPermission, Collection<? extends PermissionSubject>)`

376 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
377 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/CoreEffects.html

`StructurePage.STRUCTURE_BOARD_WITH_DETAILS` is no longer supported and has been marked `@Deprecated`.

New methods `StructureUtil.ivn(String)` and `TypeUtils.ivn(String)` convert a string to a nullable `Integer`. They complement `iv()`, `lv()`, and `lvn()` that these classes already had.

7.14.6.3 New REST API for App Configuration

There is a new REST API for working with the app configuration. It lets you read and update the list of projects for which Structure is enabled, as well as Structure's global permissions. Please refer to the documentation(see page 1121) for a detailed description.

7.14.7 Structure 6.2.1 Release Notes

2nd of September, 2020

Structure 6.2.1 is a patch release for 6.2. It adds Jira 8.12 compatibility and a security fix.

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[378] (no installation required)

7.14.7.1 Patch Release

Structure 6.2.1 is a patch release for Structure 6.2. It adds Jira 8.12 compatibility and a security fix

7.14.7.2 Supported Versions

Structure 6.2.1 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Desk).

Compatible plugins versions:

- Structure.Gantt[379] 2.4+
- Structure.Pages[380] 1.5+
- Structure.Testy [381] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

378 https://demo-structure.almworks.com/
379 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
380 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
381 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.14.7.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [382]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[383] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 56).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

382 https://wiki.almworks.com/display/structure2gmaster/Download
383 https://wiki.almworks.com/display/structure2gmaster/Download

7.15 Structure 6.1 Release Notes

3rd of June, 2020

Structure 6.1 adds rich text to Memos, the ability to use JQL/S-JQL and text filters in formula, support for Jira Anonymization and more

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[384] (no installation required)

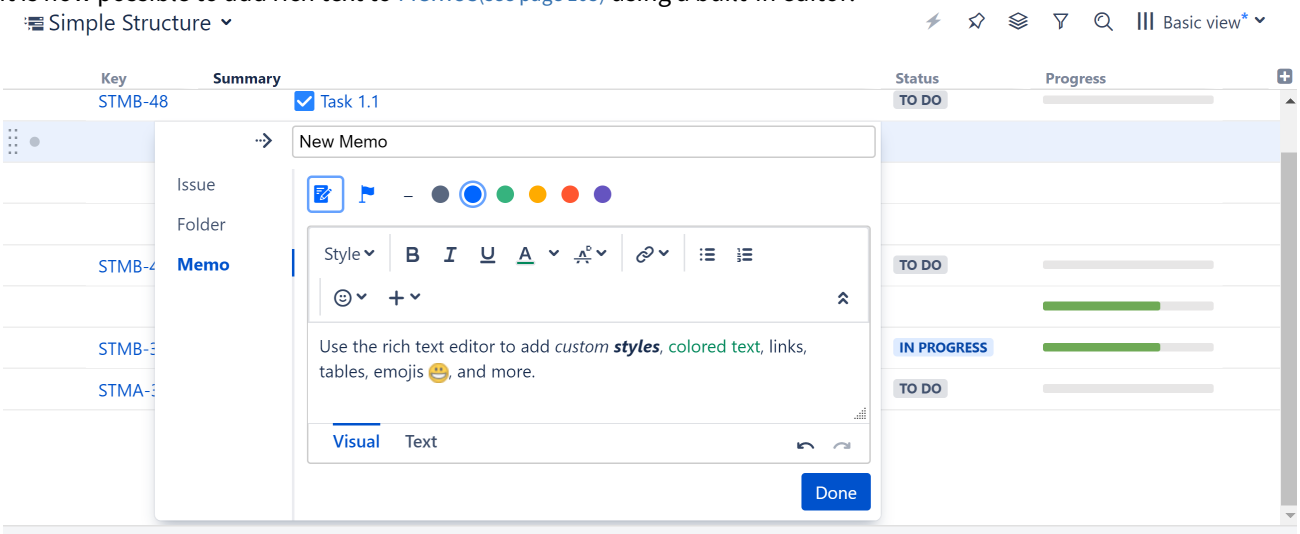
7.15.1 Version Highlights

- Rich text editor for Memos
- JQL, S-JQL and Text Query Match columns
- JQL, S-JQL and Text filters in formulas
- Support for Jira Anonymization

7.15.2 Changes in Detail

7.15.2.1 Rich Text Editor for Memos

It is now possible to add rich text to Memos(see page 105) using a built-in editor.



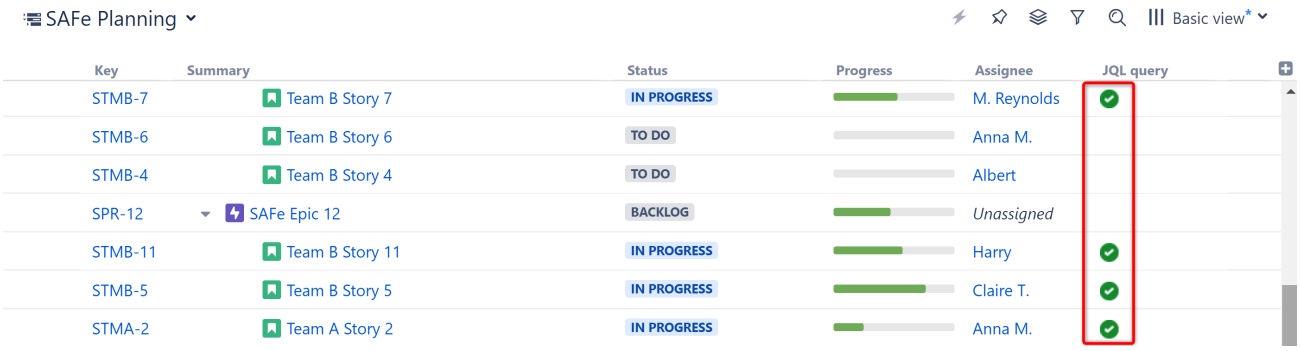
You can also edit text using wiki markup by switching to the Text mode - see Jira's Text Formatting Notation Help page[385] for a complete list of available formatting options and conventions.

Documentation: Memo(see page 105)

384 https://demo-structure.almworks.com/
385 https://jira.atlassian.com/secure/WikiRendererHelpAction.jspa?section=all

7.15.2.2 Query Match Columns

Query Match columns allow you to mark issues that match a specified JQL, S-JQL(see page 428) or text query. Items that match the query are noted by a check mark; those that do not are left blank.

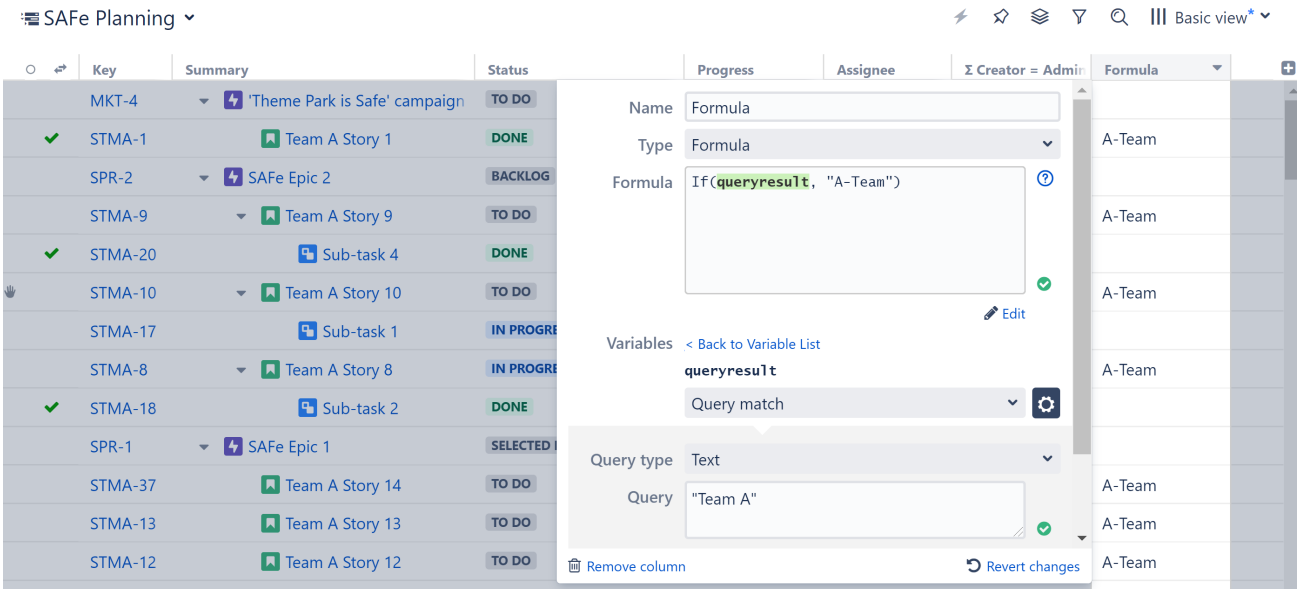


It is also possible to return a numeric value of "1" for any issue matching the query, so you can aggregate your results up the hierarchy.

Documentation: Query Match Column(see page 486)

7.15.2.3 JQL, S-JQL and Text Filters in Formulas

Using Query match, you can now filter formula results based on a JQL, S-JQL(see page 428) or text query.



Documentation: Query Match as Variables(see page 765)

7.15.2.4 Jira Anonymization

Structure now supports Jira Anonymization. When an admin anonymizes a user, any private structures, views, Effector processes or statistics owned by the user will be deleted. Shared structures or views owned by the user will be transferred to another user.

Documentation: Anonymizing Users(see page 1013)

7.15.2.5 Additional Updates

- Issue summary now appears in the header of the Issue Details panel, so it remains visible as you scroll
- Epic link can now be defined when cloning a structure
- Fixed: the keyboard shortcut "i" (assign to current user) did not work on Agile Boards
- Fixed: Epic Sum Up custom field values were displayed incorrectly within structures

7.15.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.15.3.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [386] page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.15.3.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.0 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.15.3.3 Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

386 https://wiki.almworks.com/display/structure2gmaster/Download

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[387] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Structure Quick Start Guide(see page 765).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.15.4 Enterprise Deployment Notes

In this release we have further optimized caching in the attribute system. Now in certain scenarios, when there are many structures that are not frequently used, Structure will clear its attribute caches faster, returning memory to the JVM. There are no particular special areas of interest for load testing and stress testing Structure 6.1. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[388].

7.15.5 API Changes in Structure 6.1

7.15.5.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.2.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.15.5.2 Compatible Changes in the Java API

`StructureDeletedEvent`

Starting with version 6.1, Structure will fire a `StructureDeletedEvent` to Jira's `EventPublisher` whenever a structure is deleted. You can register a listener for this event, for example, if you need to delete any extra data that your code associates with a particular structure.

387 https://wiki.almworks.com/display/structure2gmaster/Download
388 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

New utilities

We have added two new classes to the `com.almworks.jira.structure.api.util` package. Both classes are related to integers[389], our primitive collections library:

- `IntegersUtil` contains two methods extracted from `StructureUtil` (to be used in the cloud-based version of Structure).
- `JavaListToLongListAdapter` lets you represent a regular Java `List` as a `com.almworks.integers.LongList` using a `ToLongFunction` for conversion.

7.15.6 Structure 6.1.1 Release Notes

17th of June, 2020

Structure 6.1.1 fixes Advanced Wiki Markup and adds Advanced Roadmaps 3.29 support

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[390] (no installation required)

7.15.6.1 Patch Release

Structure 6.1.1 fixes an issue with Advanced Wiki Markup in formulas and descriptions, and adds Advanced Roadmaps (formerly Portfolio) 3.29 support.

7.15.6.2 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [391]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

389 https://bitbucket.org/almworks/integers
390 https://demo-structure.almworks.com/
391 https://wiki.almworks.com/display/structure2gmaster/Download

Upgrading Structure

The upgrade procedure from versions 3.0–6.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[392] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Structure Quick Start Guide(see page 769).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.15.7 Structure 6.1.2 Release Notes

6th of July, 2020

Structure 6.1.2 fixes an issue with Jira 8.10 compatibility

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[393] (no installation required)

7.15.7.1 Patch Release

Structure 6.1.2 fixes a compatibility issue with Jira Data Center starting from version 8.10. The fix is related to the way Structure issue events are dispatched in Data Center (mostly) and Server. Upon upgrading, Structure will remove the old Structure Index Monitor custom field and create a new global custom field with the same name and description.

392 https://wiki.almworks.com/display/structure2gmaster/Download
393 https://demo-structure.almworks.com/

Upgrade is highly recommended for Jira 8.10+

7.15.7.2 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [394]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[395] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Structure Quick Start Guide(see page 770).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

394 https://wiki.almworks.com/display/structure2gmaster/Download
395 https://wiki.almworks.com/display/structure2gmaster/Download

7.16 Structure 6.0 Release Notes

26th of March, 2020

Structure 6 adds Effectors, user-based permissions, and API changes.

Download the latest version of Structure and its Extensions(see page 1164)

Try It: Structure Sandbox Server[396] (no installation required)

Pricing Changes

We are introducing new pricing for Structure Server, effective April 1, 2020. Learn more:
2023-04-26_13-51-52_Structure for Jira New Server Pricing(see page 772)

7.16.1 Version Highlights

- Effectors - a new form of automation for writing Structure attributes to Jira fields
- Possibility to define permissions per user
- API changes[397] that may affect Structure integrations
- Attribute sensitivity settings
- Improved rank sorting

7.16.2 Changes in Detail

7.16.2.1 Effectors

The Attribute to Issue Field Effector allows you to write the values from Structure attributes (formulas, structure-specific columns, Structure.Gantt attributes, etc.) to Jira issue fields.

Sorry, the widget is not supported in this export.
But you can reach it using the following URL:

https://www.youtube.com/watch?v=QSreuWUQTp8

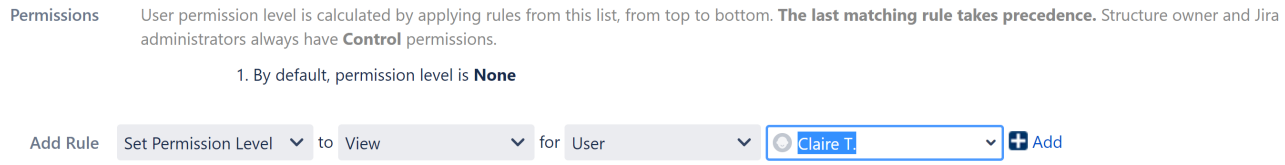
For more information, see our Effectors documentation(see page 179) and check out these sample use cases:

- Calculate Epic Story Points Based on Sub-issues(see page 637)
- Update Assignees to Match Parent Issues(see page 664)
- Calculate the Cost to Complete Issues and Projects - and Write Those Costs to Jira(see page 642)

396 https://demo-structure.almworks.com/
397 https://wiki.almworks.com/display/structure2gmaster/Structure+6+API+Changes

7.16.2.2 Permissions Per User

It is now possible to assign structure permissions to specific users. Previously permissions could only be set based on user groups or project teams.



Documentation: Structure Permissions(see page 543)

7.16.2.3 API Changes

Structure 6 introduces significant changes to Java API. (REST API remains backwards compatible.) The changes are called for because of a large remake of the attributes subsystem in the product.

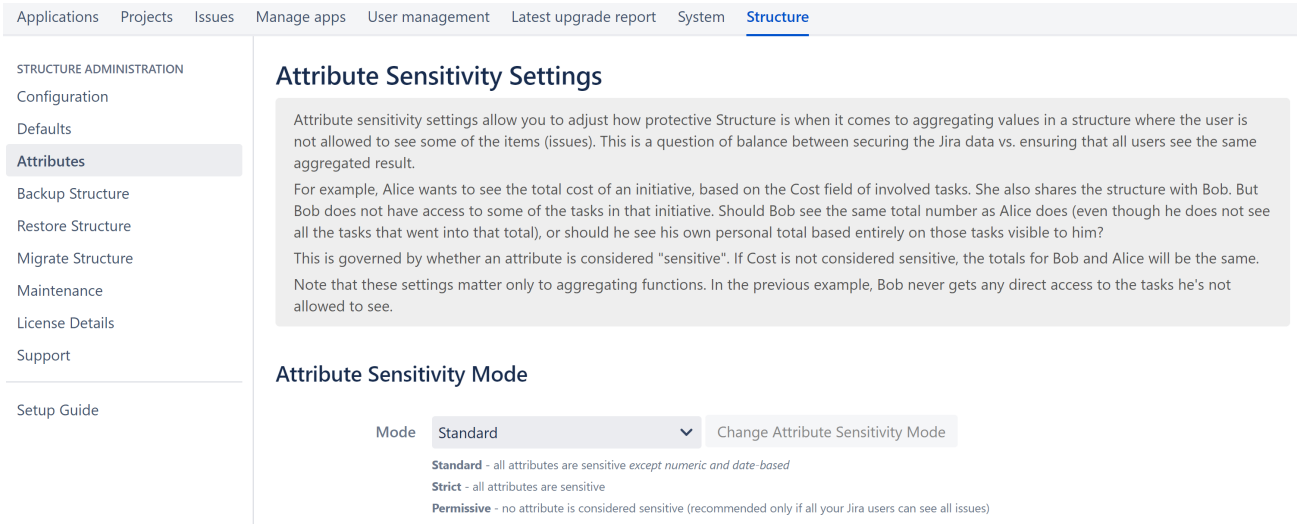
If you have an integration or an extension of Structure, your code might break. More specifically:

- If you're creating your own attribute loaders, your code will definitely be incompatible and will need some work.
- If you're just using the attribute subsystem, such as `StructureAttributeService`, your code may well break, but fixing it will most likely be trivial.
- If you're using REST APIs or other parts of Structure API, your code is very unlikely to stop working.

For more specifics about these API changes, see Structure 6 API Changes(see page 777).

7.16.2.4 Attribute Sensitivity Settings

Attribute sensitivity settings provide a new level of data security within Structure, allowing admins to specify fields that may contain sensitive information and, therefore, should not be included when calculating aggregated functions.



Documentation: Attribute Sensitivity Settings(see page 960)

7.16.2.5 Improved Rank Sorting

When items are sorted by rank, sub-issues now appear beneath their parent items.

7.16.3 Supported Versions

Structure 6.0 and all extensions support Jira versions 7.13 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

Compatible plugins versons:

- Structure.Gantt[398] 2.3
- Structure.Pages[399], 1.5
- Structure.Testy [400] 2.4

7.16.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.16.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [401]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.16.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

---

398 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
399 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
400 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
401 https://wiki.almworks.com/display/structure2gmaster/Download

7.16.4.3 Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[402] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Structure Quick Start Guide(see page 772).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.16.5 Enterprise Deployment Notes

Structure 6.0 is a major release with a lot of changes in the attribute system and an important new feature, effectors.

7.16.5.1 Changes to the Attribute System

In Structure 6.0 we have largely rewritten the attribute system, which is a crucial part of Structure, responsible for displaying all values that you see in columns inside the Structure grid, including issue field values, aggregates, and formulas. We have run extensive performance tests internally, and we are pretty sure that the new attribute system performs no worse than the old one, and significantly better in many cases.

However, we still advise you to perform load- and stress-testing on a staging environment before you upgrade, especially if you rely on aggregates and complex formulas, or export large structures to printable and Excel formats.

7.16.5.2 Effectors

Effectors are a completely new feature, and they shouldn't affect the performance of the rest of the product. However, effectors were a pretty popular feature request, and we expect their wide adoption. If you already have a potential use case for effectors in your organization, we advise you to try them on a staging system before upgrading.

Using effectors can put a noticeable load on the system, because running an effector, especially on a large structure, could require a lot of issues to be loaded from the database, and then updated. Because of that, we designed effectors so that they do not run automatically; instead, they have to be controlled manually by Structure users. Effectors always operate on behalf of the user who runs them, and all the necessary permissions are checked as if that particular user were trying to make the changes.

402 https://wiki.almworks.com/display/structure2gmaster/Download

Access to Effectors

By default, any user who has the "Access Automation" global Structure permission will be able to add effectors to their structures, and any user with the "Bulk Change" global Jira permission will be able to run previously-installed effectors. If you prefer a gradual roll-out, you can limit users' access to effectors as described in Changing Permissions to Configure and Run Effectors(see page 956).

Effector Execution

Running an effector is a two-step process. First, a preview is generated, listing all the changes that the effector is going to make. Then, if the user approves the preview, those changes are actually made, in a separate background process.

To calculate effector previews, Structure uses a separate thread pool (in a Data Center environment, on each node), so several previews can be calculated in parallel. By default, the effector thread pool is limited to N+1 threads, where N is the number of available processor cores. You can change the maximum size of the thread pool by setting the "structure.jobManager.effectorThreads" dark feature.

In contrast to preview calculation, effectors never update issues in parallel. If several users want to make changes using effectors, their requests are queued and processed one-by-one. In a Data Center environment, a single node is responsible for making the changes. If that node leaves the cluster, a different node picks up this responsibility, continues the current task from where it stopped, and takes the next task from the queue.

New Database Tables

There are three new tables in the database schema to store the data related to effectors.

The `AO_8BAD1B_EFFECTOR_INSTANCE` table stores effector parameters. A new row is created each time a user adds an effector to a structure. The data from this table is exported when you back up Structure data and can be migrated to a different instance. The table is also exported along with all Structure data during Jira backup.

The `AO_8BAD1B_EFFECTOR_PROCESS` table stores current and past effector runs. There is one row per process, regardless of how many effectors are run. When a preview is calculated, it is also stored in this table, and deleted when all changes are applied. Effector runs are not backed up by Structure, but the table is exported during Jira backup.

The `AO_8BAD1B_EFFECTOR_RECORD` records all changes made by effectors, and how to undo them. There is one row for each update attempt, successful or not. This data is not backed up by Structure, but the table is exported during Jira backup.

There is a new maintenance task that deletes old effector processes and their changes. By default, finished effector processes and their changes are kept for 30 days, and unfinished processes that didn't make any changes are kept for 24 hours. For more information on automatic maintenance please refer to the documentation(see page 966).

Need help or have questions? Contact Tempo Support[403].

403 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.16.6 Structure 6 API Changes

7.16.6.1 State of the API

With Structure 6, we are introducing a number of breaking changes to Java API. (REST API remains backwards compatible.) The changes are called for because of a large remake of the attributes subsystem in the product.

If you have an integration or an extension of Structure, your code might break. More specifically:

- If you're creating your own attribute loaders, your code will definitely be incompatible and will need some work.
- If you're just using the attribute subsystem, such as `StructureAttributeService`, your code may well break, but fixing it will most likely be trivial.
- If you're using REST APIs or other parts of Structure API, your code is very unlikely to stop working.

7.16.6.2 Conceptual API Changes

Attribute Sensitivity Configuration

There are new global per-instance settings in Structure, called attribute sensitivity configuration. The administrator has access to them via Administration | Structure | Attributes.

These settings govern the situation where a total or some other type of aggregation is performed over a forest that the user does not have a full access to – there are items that they don't see.

StructureConfiguration[404] interface now can be used to read or update these settings with `getAttributeSensitivitySettings()` / `setAttributeSensitivitySettings()`.

Attribute Value Trails Removed

While the concept of trails remains, the API clients no longer have to deal with the trails. The clients are not supposed to implement their own cache – instead, they can request loading as frequently as needed, or use the Attribute Subscription Service.

Asynchronous Attribute Loading

`AttributeSubscriptionService` is now a preferred way to continuously receive updates on the attribute values for a particular set of rows and attributes. It is being used by the Structure grid and through REST APIs.

Support for Super-Root

We have introduced a concept of a "super-root", which is a virtual row above all the roots in the forest. It can be used to calculate a total across the whole forest, for example.

In order to support the super-root we had to reserve `-1` as a special row ID number that identifies the super-root and only the super-root. It's now impossible to use `-1` in Forest structures.

For more details, see `SuperRootRow` class.

---

404 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/settings/StructureConfiguration.html

7.16.6.3 Detailed API Changes

We list only some of the most important changes in this list. For a detailed description of the current API, see the javadocs(see page 1069).

Class Name	Backwards-Incompatible Changes	Other Changes
`AttributeValue`	• Moved to package c.a.j.s.api.attribute. • Trails are no longer a part of the value.	• Stricter contract on nulls is defined.
`StructureAttributeService`	• Method `getAttributeValues()` now returns `RowValues` instead of `VersionedRowValues`. • Method `getAttributeValues()` with both `Forest` and `ForestSpec` parameters is removed.	• `loadAttributeValues()` methods, which push values through a receiver interface, are added. • `getConsistentAttributeValues()` methods are added. • `getItemValues()` method is added – it can load some attributes just for the item IDs, without a forest. • `isItemAttribute()` method helps with figuring out if an attribute is item-based or forest-based.
`ValueFormat`	• Method `cast(AttributeValue)` is removed.	
`AttributeSpec`	• Method `cast(AttributeLoader)` is removed.	• Attribute normalization rules are updated. Technically, it's a breaking change, but it should not affect custom attributes.
`RowValues`	• `getTrail()` method removed.	• `consume()` method added.
`RestAttributeSpec`	• `fromModel()` method renamed to `toRest()`.	
`ItemForestBuffer`	• Class is removed.	
`MigrationAwareSynchronizer`	• Class is removed.	
`SyncUtil`	• Class is removed.	

7.16.6.4 Conceptual SPI Changes

You only need to bother with SPI if you extend Structure functionality by adding new attributes.

New Attribute Loader Hierarchy

With the new Attributes subsystem we've revamped and extended the types of attribute loaders / attributes that are supported. The new Structure now has support for the following attributes:

- Item attributes are based only on an item and not on its position in the forest.
- Derived attributes are calculated only based on the values of other attributes.
- Single-row attributes are calculated based on the item and the row in the forest that this item is in.
- Multi-row attributes are all based on the dependencies and a particular type of forest-based aggregation.
    - Aggregate attributes are based on the children values of the same attribute.
    - Propagate attributes are based on the ancestor value of the same attribute and sibling rows.
    - Scanning attributes are based on the value of a preceding row, without regard for the hierarchy and when the hierarchy is fully expanded.

For a more in-depth description, see the AttributeLoader javadoc page[405]. Note that there's a parallel hierarchy of `AttributeLoaderContext` classes that has changed.

Attribute Caching

The new attribute subsystem is capable of caching attribute values in much more cases than the previous one. It is suggested that the implementations of attribute loaders avoid using `AttributeCachingStrategy.MUST_NOT` and review the `AttributeLoader` interface to see how the caching may be supported, for example, through context dependencies.

Also, we have introduced a new caching strategy – `AttributeCachingStrategy.SHOULD_NOT`, which is handy when the calculated object is heavy and should not be stored, but it allows caching of the dependent attributes.

Context Dependencies

The loaders may now declare their dependencies on some contextual values, for example, on user locale. This governs the caching and invalidation.

On the other hand, in order to call methods like `AttributeLoaderContext.getLocale()`, the loader must declare the context dependency.

Loader Builders

Item and single row loaders can now be built more conveniently with a number of builders. Start with `AttributeLoaders` class.

7.16.6.5 Detailed SPI Changes

We list only some of the most important changes in this list. For a detailed description of the current API, see the javadocs(see page 1069).

405 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/loader/AttributeLoader.html

Class Name	Backwards-Incompatible Changes	Other Changes
`AttributeContext`	• Replaced method `getBaseForestSpec()` with `getBaseStructureId()`	• Added `getI18n()` method • Added `getTimezone()` method
`AttributeLoader`	• Replaced the sub-interfaces and their methods, including how values are loaded	
`AttributeLoaderProvider`	• `createAttributeLoader` method now accepts `AttributeProviderContext` instead of `AttributeContext`	
`BulkAttributeLoader`	• `BulkAttributeLoader` interface is eliminated. Every loader (except DerivedAttributeLoader) is now a bulk-loading loader.	
`AbstractDistinctAggregateLoader`	• Renamed to `AbstractNaiveDistinctAggregateLoader`	
`AttributeLoaderSecurity`	• Class removed: there's no need in supporting value security in the individual loaders anymore – the system does it by itself, with the sensitivity settings governing the multi-row loaders.	
`AbstractDistinctSumLoader`	• Distinct sum loaders are currently removed from the API.	
`SimpleAttributeProvider`	• Moved to package `c.a.j.s.api.attribute.loader.basic`.	
`TrailItemSet`	• Moved to package `c.a.j.s.api.attribute.loader`.	

7.16.6.6 Effector APIs and SPIs

Structure 6.0 also adds a set of new APIs and SPIs related to effectors. This section only provides a basic overview, for details please refer to the Javadoc[406].

406 https://almworks.com/structure/javadoc/latest/

Managing Effectors

You need the `EffectorInstanceManager`[407] to create, retrieve, update, and delete *effector instances*. To create an effector instance, you must provide a complete module key of an effector implementation and a map of parameters. Once you have the effector instance ID, pass it to `CoreIdentities.effector(long)`[408] to get an item identity, which you'll be able to insert into a structure.

As of version 6.0, Structure provides only one effector implementation, the Attribute to Issue Field(see page 179) effector. Its module key is "com.almworks.jira.structure:effector-attribute-to-field".

Running Effectors

You need the `EffectorProcessManager`[409] to run effectors. You start an *effector process* by calling `startEffectorProcess()`, which returns a process ID. You then use this process ID to track the status of your process, confirm effect application, and examine the results.

For a finished process, you can also start a new process that would undo the changes it made.

Writing a New Effector

As a Jira plugin developer, you can extend Structure with new effectors. Structure 6.0 adds two new extension points, *effectors* and *effect providers*. Effectors look at a forest and attribute values to generate *effect descriptions* (represented by `StoredEffect`[410]). Effect providers validate effect descriptions and perform the requested changes.

This architecture supports code reuse. For example, Structure itself knows how to properly update most Jira issue fields, and supplies corresponding effect providers. So if all your effector does is update issue fields, you can use those providers and avoid writing issue-changing code yourself.

Effectors implement the `Effector`[411] interface and are registered using the `<structure-effector>`(see page 1082) module type. Your effector implementations can use the static factory methods in the `CoreEffects`[412] class to generate effect descriptions that will be handled by Structure.

Effect providers implement the `EffectProvider`[413] interface and are registered using the `<structure-effect-provider>`(see page 1083) module type.

7.16.7 Structure 6.0.1 Release Notes

5th of May 2020

Structure 6.0.1 provides performance and security related fixes

407 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/instance/EffectorInstanceManager.html
408 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/CoreIdentities.html
409 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/process/EffectorProcessManager.html
410 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/StoredEffect.html
411 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/Effector.html
412 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/CoreEffects.html
413 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/EffectProvider.html

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[414]

7.16.7.1 Patch Release

This is a small patch release based on Structure 6.0. It provides performance and security related fixes.

Upgrade is recommended for all customers using Structure 6.0.

7.16.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.16.7.3 Upgrade

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 6.0.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.16.7.4 Enterprise Deployment Notes

Structure 6.0.1 introduces attribute caching optimizations for structures with pinned items.

This is designed to reduce memory consumption for structures which are loaded on Issue View pages because they contain pinned issues. Since structures are not shown on the Issue View pages of issues added through Automation, this is particularly important for instances with huge structures built by synchronizers or containing large amounts of static content.

7.17 Structure 5.6 Release Notes

> ⓘ **19th of September, 2019**
>
> Structure 5.6 adds column pinning, cache clearing, and more.

414 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Download the latest version of Structure and its Extensions
(see page 1164)Try It: Structure Sandbox Server[415] (no installation required)

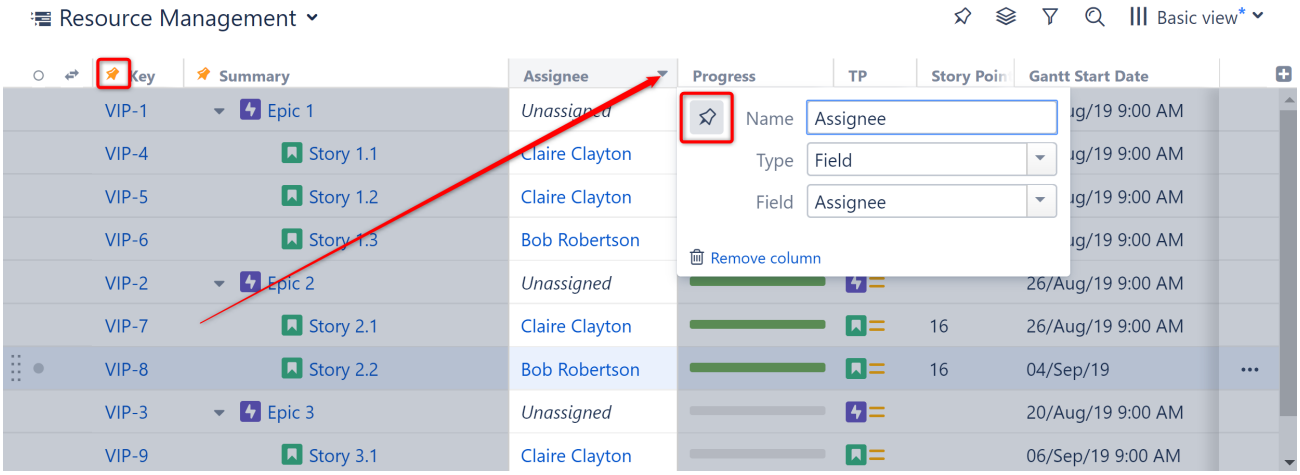
7.17.1 Version Highlights

- Columns can be "pinned" to remain visible when horizontal scrolling is enabled.
- Structure caches can now be cleared for users and nodes from within the app.
- Adds support for Jira 8.4

7.17.2 Changes in Detail

7.17.2.1 Column Pinning

When Horizontal scrolling is enabled, the Summary column is "pinned" so that it will always be visible as users scroll through other columns. Now users can pin additional columns, so they will remain visible as well.



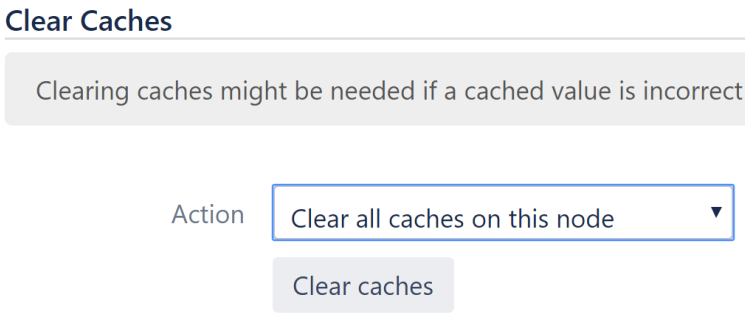
Pinned columns will be moved to the left side of the screen and will remain visible as you scroll.

Documentation: Horizontal Scrolling(see page 503)

7.17.2.2 **Clearing Structure Caches**

Structure caches can now be cleared from within the app for specific users or the whole Jira instance. In Jira Data Center, it's possible to clear caches for the current node or all nodes in the cluster.

415 https://demo-structure.almworks.com/



Documentation: Clearing Structure Caches(see page 1012)

7.17.2.3 Additional Updates

- Archived structures are now clearly marked as archived when opened.
- When viewing a structure on a Project page, the current project is auto-selected when adding a new issue.
- Fixed: Compatibility with Jira add-ons, "Links Hierarchy - Core, Epic & Portfolio" and "Tree CustomField"

7.17.3 Supported Versions

Structure 5.6 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.5. Structure.Testy[416]extension, Colors[417], Structure.Pages[418], Structure.Gantt[419]and integrations with third-party apps should continue working normally.

7.17.4 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.17.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 1164) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.

416 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
417 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
418 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
419 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.17.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure from versions 3.x–5.5 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.17.5 Enterprise Deployment Notes

Structure 5.6 contains a workaround for a performance problem that could affect extremely large Jira 7 instances, and a few other changes in important areas such as client-server communication, structure generation and locking, and user action handling.

7.17.5.1 Lucene Searcher Flushing

Jira uses Apache Lucene as the search engine behind JQL, and so Structure also relies on Lucene for JQL and a variety of other tasks, such as checking users' access to issues and reading issue data quickly. To work with Lucene, an app uses a Lucene searcher object provided by Jira. Lucene searchers are shared between Jira itself and all apps; when an issue is updated and re-indexed, the current Lucene searcher is released, and it can be garbage-collected when all tasks still using it are finished.

In Jira 7, on extremely large instances, Lucene searchers and associated data structures can consume huge amounts of memory – see JRASERVER-67805[420], JRASERVER-68439[421]. This shouldn't affect Jira 8, which uses a newer version of Lucene, where these problems are fixed.

We have seen a few support cases, with very large Jira 7 instances, where a long-running Structure task would hold on to a Lucene searcher that consumes several gigabytes of memory. This would prevent the JVM from garbage-collecting the searcher and reclaiming that memory, which in turn would negatively affect the performance of the whole instance or cluster node. Note that Structure itself does not necessarily *need* all that memory; the shared searcher might get inflated by an unrelated task and *then* be given to Structure.

Lucene searcher flushing is a new, experimental dark feature introduced to work around this problem. If it's enabled, Structure will release ("flush") the Lucene searcher not at the end of a task, but immediately after using it. The JVM will then have a chance to garbage-collect the searcher without waiting for the current Structure task to finish.

Lucene searcher flushing is available only on Jira 7. It is enabled automatically for instances where the number of issues in the index multiplied by the number of searchable fields is greater than or equal to 2^{30} (1,073,741,824). It can also be enabled or disabled manually by setting the `structure.lucene.flushSearchers` dark feature to "true" or "false", respectively. If you are running Jira 7, and your instance is above the threshold or close to it, and you have large structures that take a long time to generate, we advise that you load- and stress-test your instance before upgrading.

---

420 https://jira.atlassian.com/browse/JRASERVER-67805
421 https://jira.atlassian.com/browse/JRASERVER-68439

7.17.5.2 Reduced HTTP Thread Pool Usage

Structure does not perform potentially costly operations like structure generation in HTTP threads, offloading them to background thread pools instead. In previous versions of Structure, an HTTP thread would start a background job, wait for it to finish, and if it's not done in 10 seconds, return the job ID to the client. However, the client would immediately re-submit the request, and thus it would effectively always keep one HTTP thread busy for the duration of the background task.

As we have observed in a recent support case, on an overloaded Jira instance, Structure's background tasks could also slow down, and, in extreme cases, too many HTTP threads would be busy waiting for Structure tasks, making it hard for the instance to serve other HTTP requests.

To reduce chances of HTTP thread pool starvation, Structure 5.6 introduces client-side HTTP timeouts. When the client receives a job ID, it will wait 1 second before re-submitting the request, then 2 seconds, 4 seconds, and so on, up to 1 minute. We have also reduced the server-side timeout from 10 seconds to 5 seconds, so HTTP threads are returned to the pool sooner, improving the throughput.

7.17.5.3 Clearing Caches and Hard Refresh

In previous versions, hard-refreshing a page in the browser would cause Structure to clear its caches for the current user. We have decided to change this for two reasons. First, clearing the caches too often could negatively affect performance. Second, some of Structure's important caches are global, not per-user, and there was no easy way to clear them. Structure 5.6 introduces a new way of clearing caches. Hard-refreshing the page no longer works, so if you relied on it for troubleshooting Structure issues, please switch to the new interface(see page 1012).

7.17.5.4 Testing on Staging Environment

We advise you to perform load and stress testing in a staging environment before you upgrade. Given the changes in this version, you can try the following scenarios:

- Find or create a large automated structure that takes significant time to generate (around 30 seconds or more). Emulate multiple users opening it at the same time.
- Emulate multiple users making changes to Jira issues in this structure at a significant rate (say, one update every 1 or 2 seconds). Check that the structure re-generates to reflect the changes.
- Open a large manually-created structure containing 10,000 issues or more. Try moving one or more rows in this structure, note the time it takes to handle the moves.

While running these experiments, watch the log files for errors and warnings, and monitor heap usage and GC activity.

Need help or have questions? Contact Tempo Support[422].

422 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.17.6 API Changes in Structure 5.6

7.17.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.16.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.17.6.2 Compatible Changes in the Java API

Pinned columns in `ViewSpecification`

With the introduction of pinned columns in Structure 5.6, we have added a list of pinned column IDs to the view specification. There are 3 new methods to work with the list:

- `ViewSpecification.getPins()`
- `ViewSpecification.Builder.setPins(List<String>)`
- `ViewSpecification.Builder.getPins()`

Resetting the `ItemTracker`

We have added a new method named `reset()` to the `ItemTracker` interface. It's meant to be used in situations when there are too many changes, or when it's impossible to know which items have changed. All clients calling `getUpdate()` after the `reset()` call will receive full updates.

7.17.7 Structure 5.6.1 Release Notes

> **14th of October 2019**
>
> Structure 5.6.1 fixes a script dependency that may cause the Issue Navigator page to load incorrectly.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[423]

7.17.7.1 Patch Release

This is a small patch release based on Structure 5.6.0.

423 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is recommended for all customers using Structure 5.6.0.

7.17.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.17.7.3 Upgrade

⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.17.8 Structure 5.6.2 Release Notes

ⓘ **24th of December 2019**

Structure 5.6.2 provides Jira 8.4 compatibility fixes and more.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[424]

7.17.8.1 Patch Release

This is a patch release based on Structure 5.6.1. It provides:

- Compatibility with Jira 8.4+.
- Fixed: Epic Link colors now appear correctly in gadget.
- Fixed: “Archived” label appeared next to search type when a search is opened.
- Fixed: After collapsing and expanding structure levels, user could not select status for the Structure.Testy add-on.
- Additional bug fixes.

Upgrade is recommended for all customers using Structure 5.6.x

424 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.17.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.17.8.3 Upgrade

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.17.8.4 4. Enterprise Deployment Notes

Structure 5.6.2 does not introduce changes that could affect performance or would justify additional load and stress testing for Enterprise deployments.

7.17.9 Structure 5.6.3 Release Notes

> ⓘ **24th of January 2020**
>
> Structure 5.6.3 provides performance improvements for Jira 8+ and the Portfolio Extender.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[425]

7.17.9.1 Patch Release

This is a patch release based on Structure 5.6.2. It provides performance improvements for Jira 8+ and the Portfolio Extender.

Upgrade is recommended for all customers using Structure 5.6.x.

425 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.17.9.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.17.9.3 Upgrade

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.17.9.4 4. Enterprise Deployment Notes

Structure 5.6.3 significantly increases Lucene index reading operation performance on Jira 8+. The new version of Lucene in Jira 8 introduced major changes to index data structures, which made its traditional read APIs significantly slower on large amounts of data. Structure 5.6.3 compensates for this, providing overall performance on Jira 8.x comparable with Jira 7.x, or even faster. We have also improved the performance of the Portfolio children extender, which now batches Lucene queries in some cases and should work faster on Jira 8+. We advise you to run Structure 5.6.3 tests on your slowest structures pre-update (for example, structures using the Child Issues (Portfolio) Extender) and compare the new times against your current Structure version.

7.17.10 Structure 5.6.4 Release Notes

> ⓘ **5th of May 2020**
>
> Structure 5.6.4 is a security patch release.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[426]

7.17.10.1 Patch Release

This is a patch release based on Structure 5.6.3. We have addressed a security issue that affects all Structure versions starting with 3.5. Problem was fixed in the latest (6.0) Structure.

426 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is recommended for all customers using Structure 5.6.x.

7.17.10.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.17.10.3 Upgrade

⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.18 Structure 5.5 Release Notes

ⓘ **18th of July, 2019**

Structure 5.5 adds performance safeguards for structure transformations and contains a few improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 1164)Try It: Structure Sandbox Server[427] (no installation required)

7.18.1 Version Highlights

- Timeout for transformations
- Possibility to back up and restore Structure configuration
- Descending order is set by default when sorting by version
- Manual adjustments improvements and bug fixes

427 https://demo-structure.almworks.com/

7.18.2 Changes in Detail

7.18.2.1 Transformation Timeout

In Structure 4.6, we introduced a feature for pausing the structure generation(see page 201) process if it takes too much time. If loading a structure with automation takes longer than allowed, the generation process would stop to avoid performance impact on Jira. This limit was not applied to transformations, which could potentially cause the same performance impact.

In this version we've introduced a mechanism that will pause the generation process if it exceeds the time limit, regardless of whether it is caused by the automation that is a part of the structure, or by the additional transformations applied on top.

7.18.2.2 **Backup/Restore/Migration Improvements**

We have added several additional options for Structure Backup/Restore/Migration operations. Starting with Structure 5.5, the Structure configuration and the dark feature settings(see page 974) will be stored not only in Jira backups but also in Structure backups. When restoring from a Structure backup, the saved Structure configuration (like the settings set by the Jira administrator) will overwrite the active Structure configuration.

When migrating Structure data, you can choose if you want to restore the general Structure permissions settings(see page 954), such as projects enabled for Structure, and if you want to restore the enabled dark features.

7.18.3 Supported Versions

Structure 5.5 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.4. Structure.Testy[428]extension, Colors[429], Structure.Pages[430], Structure.Gantt[431]and integrations with third-party apps should continue working normally.

7.18.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.18.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

428 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
429 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
430 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
431 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 1164) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.18.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure from versions 3.x–5.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.18.5 Enterprise Deployment Notes

In terms of stability and performance, this release does not bring significant changes compared to version 5.4. There are no particular special areas of interest for load testing and stress testing Structure 5.5. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[432].

7.18.6 API Changes in Structure 5.5

7.18.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.15.0

See Structure API Versions(see page 1069) for full version information and downloads.

432 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.18.6.2 Compatible Changes in the Java API

Restoring History

Restoring Structure history from a backup file can take a long time. Now you can call `BackupOperation.setRestoreHistory()` to disable restoring history even if it's present in the backup file. The new method must be called before calling `restore()`.

Pinger

`Pinger` is an `@Internal` Structure component extracted into the API for consumption by Structure add-ons. It is not intended to be used by third-party developers.

7.18.7 Structure 5.5.1 Release Notes

> Please note that if you are using Jira Data Center after September 02, 2019, you will need to switch to the Data Center license for Structure as well (no changes to the app itself are required).

> **15th of August 2019**
>
> Structure 5.5.1 provides CDN support[433] (Jira 8.3 compatibility)

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[434]

7.18.7.1 Patch Release

This is a small patch release based on Structure 5.5.0.

Upgrade is recommended for all Data Center customers using Structure 5.5.0.

7.18.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.

433 https://confluence.atlassian.com/adminjiraserver/use-a-cdn-with-atlassian-data-center-applications-974378840.html
434 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.18.7.3 Upgrade

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 5.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.19 Structure 5.4 Release Notes

> ⓘ **5th of June, 2019**
>
> Structure 5.4 adds Memo items, Status category column, the ability to track time in status and additional improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 1164)Try It: Structure Sandbox Server[435] (no installation required)

7.19.1 Version Highlights

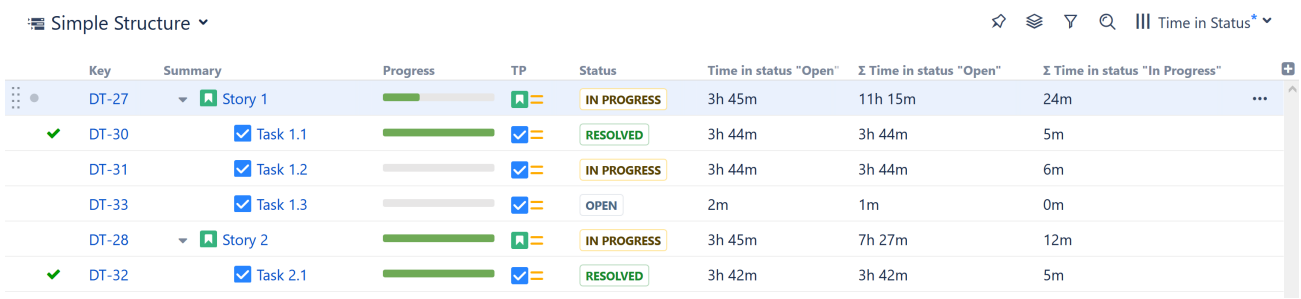
- Introduces the new Memo item, which allows you to add text notes or flags as rows within a structure
- Introduces the Status category column and the ability to use Status category in formulas
- The ability to track the time issues spend in a particular status
- Additional performance improvements and bug fixes

7.19.2 Changes in Detail

7.19.2.1 Time in Status

The Time in Status column allows you to calculate how much time issues spend in specific statuses.

435 https://demo-structure.almworks.com/

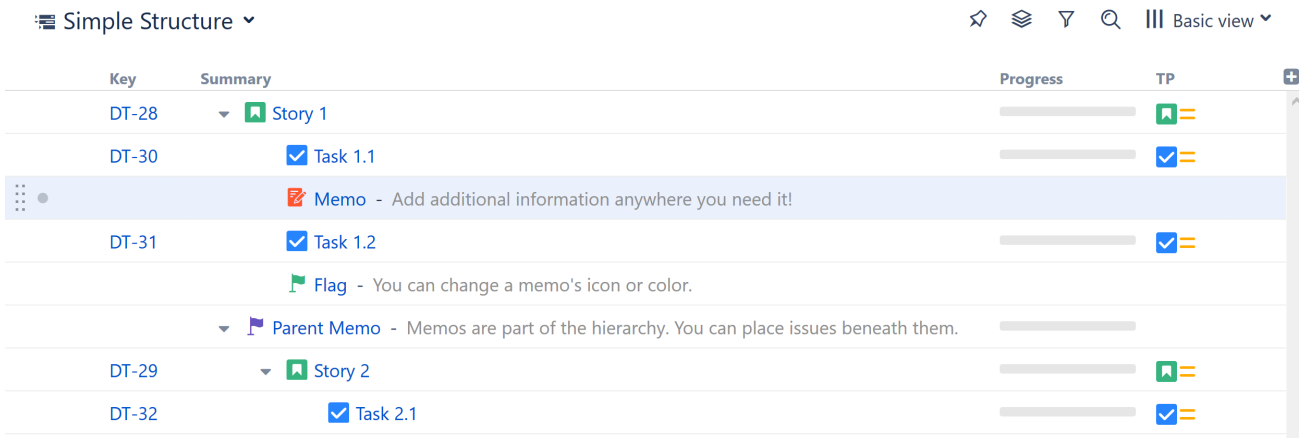


Time in Status can be customized and aggregated, depending on your business needs.

Documentation: Time in Status Column(see page 493)

7.19.2.2 Memo Items

Memo items work similar to folders within a structure, except that memos can include a choice of icons, color and text.



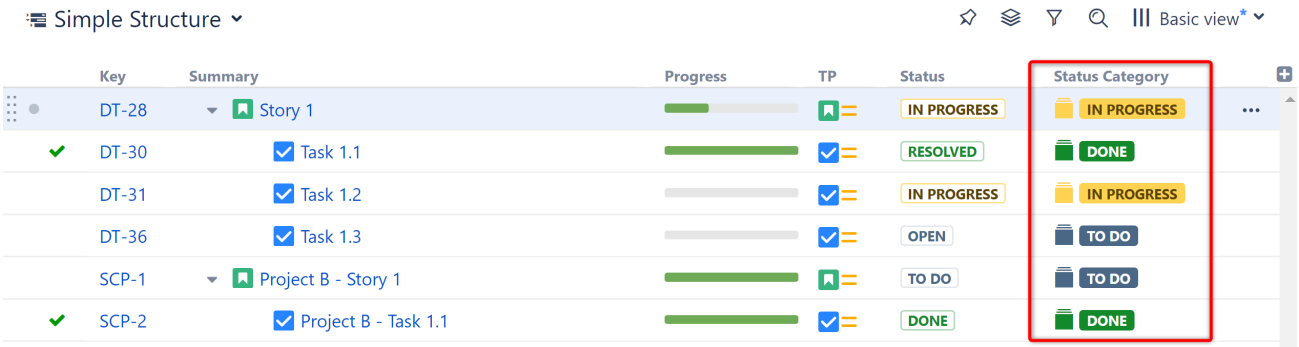
Memos can serve a variety of purposes within a structure:

- Add notes or reminders that pertain to the structure or project as a whole, rather than just a single issue (for that, try a Notes column(see page 470))
- Add high-level requirements directly to your structure
- Add a placeholder for other items
- Use them like folders, grouping issues within your hierarchy (with the added benefit of color and text)
- Just about anything else you can think of!

Documentation: Memo(see page 105)

7.19.2.3 Status Category Column

The Status category column allows you to see at a glance which Status category each issue is in. This can be extremely useful if different teams/projects use different workflows or custom statuses.



Status category values can be used to sort, filter or group issues. They can also be used in JQL queries and formulas.

Documentation: Status Category Column(see page 489)

7.19.2.4 Additional Updates

- Export to Structure has been removed from the Issue Navigator page
- Fixed: With Group by Status generators, unable to move issues within the same Status grouping
- Fixed: Summary value looks empty when editing another field
- Fixed: Values in aggregate columns (with Sum over sub-items option checked) cannot be edited after a sort
- Fixed: Summary is editable inline even if it has been removed from the edit screen
- Fixed: Power Scripts 4.6.0 breaks Structure transformations
- Fixed: Items sometime move incorrectly when Manual Adjustments are used with Automations

7.19.3 Supported Versions

Structure 5.4 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.3. Structure.Testy[436] extension, Colors[437], Structure.Pages[438], Structure.Gantt[439] and integrations with third-party apps should continue working normally.

7.19.4 Installation and Upgrade

7.19.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 1164) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.

436 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
437 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
438 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
439 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on the Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please note that there are limitations to its compatibility with Structure 4.2 and higher, so a Confluence upgrade to version 6.1 or later is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.19.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure from versions 3.x–5.3 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

> We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

7.19.5 Enterprise Deployment Notes

Structure 5.4 introduces two new features and a performance improvement important for large-scale Jira Server and Jira Data Center instances.

7.19.5.1 Periodical Cleaning of the JQL Query Literals Cache

Jira has quite a lot of request-level caches that keep expensive data easily accessible for the duration of a single user request. These caches get cleared once the request completes, and since the absolute majority of requests are quick, there is usually no problem.

However, we have seen a few support cases in which an unusually long-running background task would make a Structure thread accumulate too much data in request caches, consuming excessive amounts of memory.

We are actively working to solve this problem, and in Structure 5.4 we introduce periodical cleaning of the JQL query literals cache, which is one of the caches susceptible to the problem. Further improvements in this area are planned for future Structure releases.

If you have large structures that take significant time to generate, we advise that you check their performance in a staging environment before upgrading.

7.19.5.2 Time in Status Column

The new Time in Status column is based on issue history data, so the values can be quite expensive to calculate. Our performance tests have shown that in the aggregate mode, it can be visibly slower than simple field-based columns

in structures of 10,000 issues or more. If you are planning to use this column with large, deeply-nested structures, we advise that you check its performance on a staging system.

7.19.5.3 Memo Item Type

The new Memo item type is based on generic items, introduced in Structure 4.4. The data will be stored in the `AO_8BAD1B_GENERIC_ITEM` table, which will contain one row for each memo item. Because the table has been introduced earlier, there are no schema updates in this release.

⚠ Memos may be slower when accessing structure via Chrome with the built-in spell checker enabled.

7.19.5.4 Testing on a Staging Environment

Apart from the suggestions above, there are no special areas of interest for load testing and stress testing Structure 5.4. We advise running the same testing procedures as you've done for previous upgrades.

✓ Need help or have questions? Contact Tempo Support[440].

7.19.6 API Changes in Structure 5.4

7.19.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.14.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.19.6.2 Compatible Changes in the Java API

Memo Item Type

With the introduction of memo items in Structure 5.4, we are adding a new constant to the public API: `CoreItemType.MEMO` contains the module key for the memo item type.

Memo items were called "notes" during development, but we decided to rename them to avoid confusion with the Note column. The constant `CoreItemTypes.NOTE`, added in the previous API version, has therefore been deprecated and shouldn't be used.

---

440 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

New Methods in `TrailItemSet`

We have added two new methods to the `TrailItemSet` class:

- `fromValues()` is a static factory method that lets you combine the trails from a collection of `AttributeValues` into a `TrailItemSet`;
- `expand(TrailItemSet)` produces a set that contains all items from the receiver and the argument sets.

7.19.7 Structure 5.4.1 Release Notes

> **5th of July 2019**
>
> Structure 5.4.1 provides a number of minor fixes.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[441]

7.19.7.1 Patch Release

This is a small patch release based on Structure 5.4.0. It addresses the following issues:

- Fixed: Screen remains dark after making a status change in the Issue Details panel.
- Fixed: Empty rows appear in the structure when using a combination of Filter and Group transformations.
- Fixed: Summary data is lost when moving from the Create Issue dialogue to the Create Issue panel.
- Fixed: Cannot share Structure.Gantt chart perspectives.

Upgrade is recommended for all customers using Structure 5.4.0.

7.19.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.19.7.3 Upgrade

> If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 5.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)

441 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.20 Structure 5.3 Release Notes

26th of March, 2019

Structure 5.3 adds possibility to group by fix version name and several improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 1164)Try It: Structure Sandbox Server[442] (no installation required)

7.20.1 Version Highlights

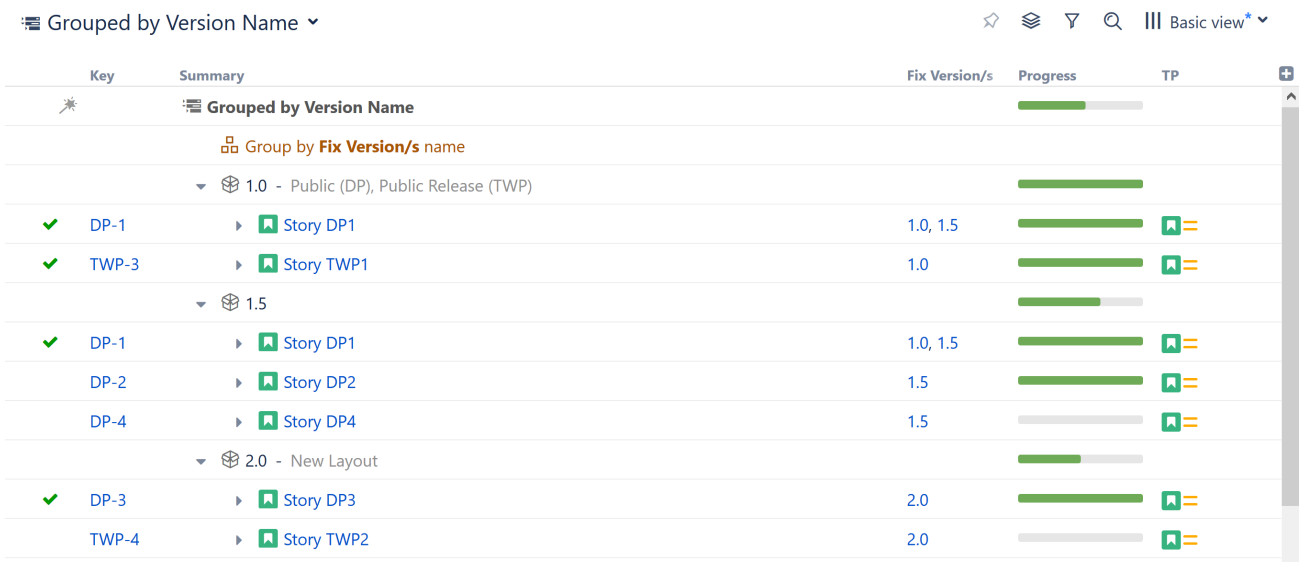
- Introduces the new Version Name Grouper, which groups issues by version, even across multiple projects
- Resolved icons (green checkmarks) are now displayed when an issue is in a Done status category, rather than anytime there is a non-blank resolution
- "Mark Manual Adjustments" option is now switched on by default when Manual Adjustments(see page 194) are enabled
- Quick Transformations panel can now be hidden while transformations remain active
- Fixed: Unable to change epic links with synchronizers installed

7.20.2 Changes in Detail

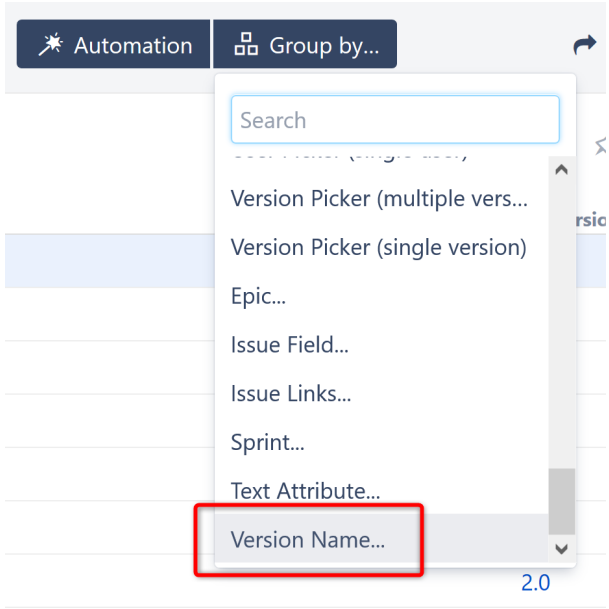
7.20.2.1 Version Name Grouper

The new **Version Name...** Grouper allows you to group issues with the same version names across multiple projects.

442 https://demo-structure.almworks.com/



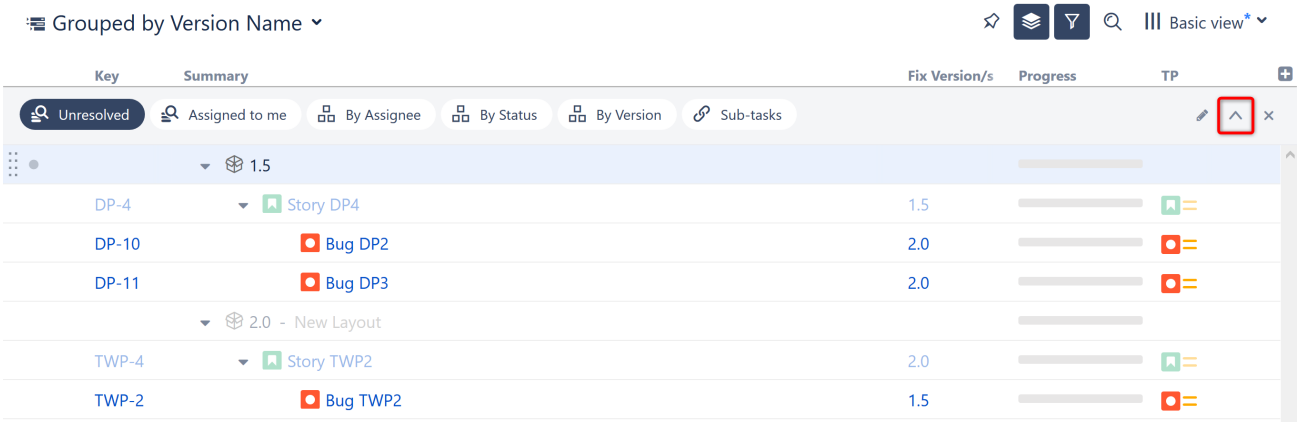
The Version Name Grouper is located at the bottom of the Group by list - or Just type "version name" into the search bar.



You can group issues by the Affects Version, Fix Version or a version custom field (multiple or single).

7.20.2.2 Hide Quick Transformations panel

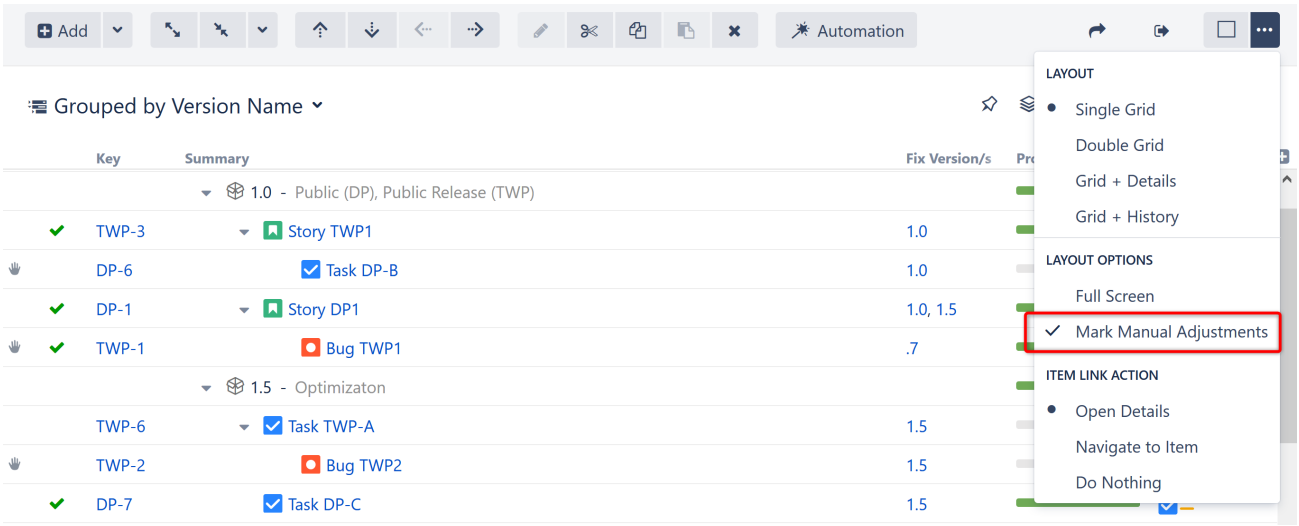
It is now possible to hide the Quick Transformations panel (as could previously only be done with the Transformations panel), while continuing to use Quick Transformations in your structure. Just click the hide button:



When transformations are applied, but the panel is hidden, the Quick Transformations button is colored blue.

7.20.2.3 Mark Manual Adjustments

The Mark Manual Adjustments option is now selected by default, whenever manual adjustments are enabled for a structure.



7.20.2.4 Additional Updates

- Resolved icons (green checkmarks) are now displayed when an issue is in a Done status category, rather than anytime there is a non-blank resolution. Admins can revert this to the old behavior using Advanced Configuration(see page 801).
- Fixed: Unable to change epic links with synchronizers installed.

7.20.3 Supported Versions

Structure 5.3 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.2. Structure.Testy[443] extension, Colors[444], Structure.Pages[445], Structure.Gantt[446] and integrations with third-party apps should continue working normally.

7.20.4 Installation and Upgrade

7.20.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 1164) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on the Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please note that there are limitations to its compatibility with Structure 4.2 and higher, so a Confluence upgrade to version 6.1 or later is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.20.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure from versions 3.x–5.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

> We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

443 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
444 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
445 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
446 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

7.20.5 Enterprise Deployment Notes

Structure 5.3 contains several changes to its core components that could be important for large installations and Jira Data Center instances.

7.20.5.1 Core Attribute Changes

The newly-introduced Version Name item type is special because its core attributes, like summary and icon, depend not on the Version Name item itself, but on the issues below it. In order to support that, we had to change how those attributes are loaded for all item types, not only Version Names.

7.20.5.2 Custom Field Scope Checks

Previous versions of Structure could show custom field values for fields that are no longer available for the project and type of the issue. Those values could also be used by generators, even though Jira itself wouldn't show or let you edit them. Starting with Structure 5.3, we check the scope of custom fields when generating structures and loading issue attributes. We use the Lucene index when possible to speed up these checks.

7.20.5.3 Testing on Staging Environment

Given the changes described above, if you have large, multi-level structures (10,000 issues or more), we advise that you test their performance on a staging system before upgrading. You can try the following scenarios:

- Open a large, multi-level structure using a view with only a few simple columns, e.g. Key, Summary, and Assignee. Note how long it takes for the field values to appear.
- Scroll down a few screens and note how quickly Structure loads field values for the new rows.
- Turn on Automation mode and see how long it takes to render the structure and generator rows.
- Add a custom field column to the Structure grid, and check how long it takes to load the values. Scroll a few screens up or down again.
- Create a large structure with a few generators based on Jira's built-in custom fields, e.g. a grouper by a User Picker field and a sorter by a Number field. Check how long it takes to generate.
- Edit an issue from the structure and change its custom field value. Note the structure's reaction time.

Watch the log files for errors and warnings while running these experiments.

The usual load and stress testing can also be performed.

Need help or have questions? Contact Tempo Support[447].

7.20.6 API Changes in Structure 5.3

7.20.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

---

447 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Jira Version	New API Version
7.6+	16.13.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.20.6.2 Compatible Changes in the Java API

Version Name Items

In Structure 5.3 we introduce a new core item type, the Version Name. It represents the canonical (lower-case, trimmed) name of a Jira version, and is used by the new Version Name grouper to combine same-named versions from multiple projects. We have added the following static methods and constants to the API to support Version Name items:

- `CoreIdentities.versionName(String)`
- `CoreIdentities.versionName(Version)`
- `CoreIdentities.isVersionName(ItemIdentity)`
- `CoreIdentities.canonicalVersionName(String)`
- `CoreItemTypes.VERSION_NAME`
- `JiraFunc.CANONICAL_VERSION_NAME`
- `JiraFunc.PROJECTCONSTANT_DESCRIPTION`

Arbitrary Parameters for Generic Items

We have added a parameter map to the `GenericItem` class (available since Structure 4.5). Parameters are stored in the database, and let you associate arbitrary information with a generic item. The following methods have been added to the API:

- `GenericItem.getParameters()`
- `GenericItem.Builder.setParameter(String, Object)`
- `GenericItem.Builder.setParameters(Map<String, Object>)`

We are currently working on a new item type, the Note, which will be based on generic items. There is a new constant for it, `CoreItemTypes.NOTE`.

Deprecated Methods in Export API

The following methods in the Export API have been deprecated:

- `ExportRequestContext.requireAttribute(AttributeSpec, boolean)`
- `ExportRow.get(AttributeSpec, boolean)`

Starting with version 5.3, Structure handles the `afterFiltering` flag for you, so you should switch to the single-argument versions of those methods.

Custom Final Step for New Structure Templates

It is now possible to skip the common final step in a new structure template wizard, if your template can automatically set the new structure name and permissions. A new default method and a constant have been added to support that:

- `NewStructureTemplate.hasCustomFinalStep()`
- `NewStructureTemplateStep.CREATE_STRUCTURE`

7.21 Structure 5.2 Release Notes

27th of December, 2018

Structure 5.2 adds support for Jira 8.0 and several performance improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 1164) Try It: Structure Sandbox Server[448] (no installation required)

7.21.1 Version Highlights

- Adds support for Jira 8.0
- Clicking the “Open” link from a Structure gadget opens Structure Board with the same filters applied
- Manage Structure is now available from the structure selection menu
- Several performance improvements and fixes

7.21.2 Changes in Detail

7.21.2.1 Support for Jira 8.0

Structure 5.2 is the first version to support Jira 8.0. We have thoroughly tested it against Jira 8.0 Beta and fixed all discovered incompatibilities.

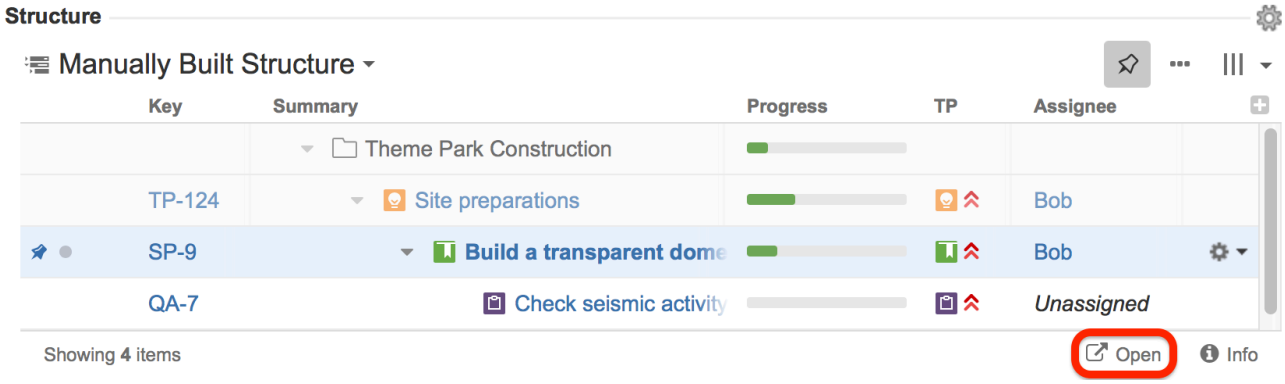
When you upgrade to Jira 8.0, your existing index files will automatically be deleted and rebuilt using Jira's new index format. During this reindexing, all structures will appear empty and read-only, and you will see a warning message about the inconsistent index. This is normal – your structures will become available again as soon as the reindexing is complete.

Additionally, background synchronization will not be possible during the reindexing. If you rely on synchronizers, you should manually resync your structures once the new index is ready.

7.21.2.2 Smoother Transition from Structure Gadget to Structure Board

When viewing a structure from anywhere other than the Structure Board, filters[449] or other transformations[450] are often applied to limit what you see to the context of that particular location. For example, when viewed from a project page, the structure is filtered to only show issues from that project. When you use the open link, those same transformations are now applied on the Structure Board.

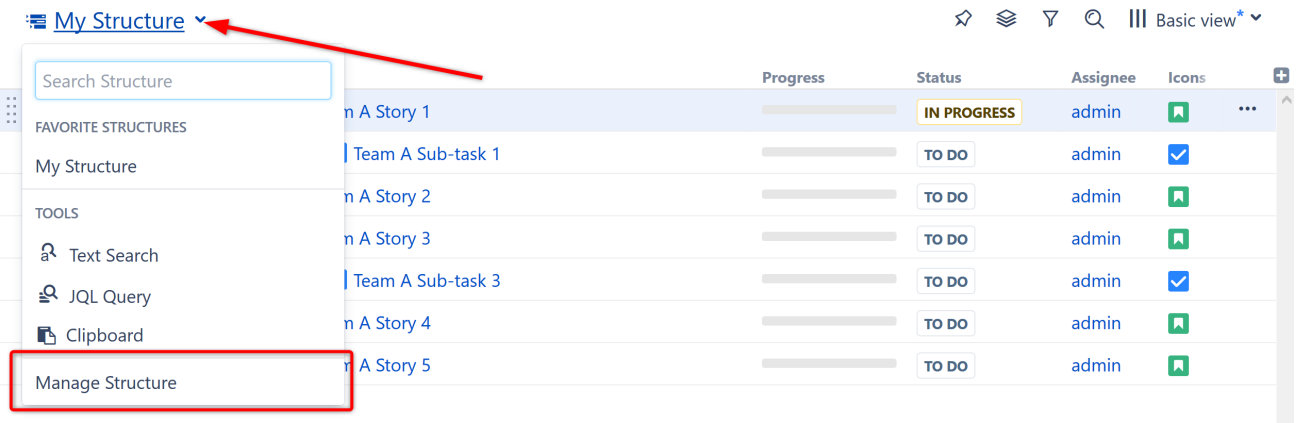
448 https://demo-structure.almworks.com/
449 https://wiki.almworks.com/display/structure2gmaster/Filter
450 https://wiki.almworks.com/display/structure2gmaster/Transformations



To review or remove the transformations, click the Transformations button in the panel toolbar.

7.21.2.3 Manage Structure in Structure Selection Menu

Manage Structure is now accessible from the structure selection menu.



7.21.3 Supported Versions

Structure 5.2 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.1. Structure.Testy[451] extension, Colors[452] , Structure.Pages[453], Structure.Gantt[454] and integrations with third-party apps should continue working normally.

451 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
452 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
453 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
454 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

7.21.4 Installation and Upgrade

7.21.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 1164) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on the Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please note that there are limitations to its compatibility with Structure 4.2 and higher, so a Confluence upgrade to version 6.1 or later is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.21.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure from versions 3.x–5.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

> We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

7.21.5 Enterprise Deployment Notes

In terms of stability and performance, this release does not bring significant changes compared to version 5.1.0, as we have mostly concentrated on compatibility, bug fixes, and minor improvements.

There are no particular special areas of interest for load testing and stress testing Structure 5.2. We advise running the same testing procedures as you've done for previous upgrades.

> Need help or have questions? Contact Tempo Support[455].

455 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.21.6 API Changes in Structure 5.2

7.21.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.12.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.21.6.2 Compatible Changes in the Java API

New methods in the `Forest` interface

We have added three new methods to the `Forest` interface and its implementation, `ArrayForest`:

- `long getNextSibling(long row)`
- `long getNextSiblingForIndex(int index)`
- `int getNextSiblingIndex(int index)`

These methods let you find the next sibling of a row. They complement the existing methods for finding the preceding sibling.

Data Center-specific licenses

We have updated the licensing API to distinguish Jira Data Center-compatible licenses from regular server and legacy licenses. The additions include:

- `LicenseData.isDataCenterCompatible()` method checks the license is compatible with Jira Data Center;
- `LicenseData.getLicenseHosting()` method returns an instance of the new `StructureLicenseHosting` enum, which tells whether this is a Data Center license, a regular server license, or a "legacy" license.
- `StructureLicenseError.NO_DATACENTER_SUPPORT` is a new error code reported when an incompatible app license is used on a Data Center instance.

7.21.6.3 JavaScript API for adding item details support

We have added a JavaScript API that allows app developers to implement item details support for specific item types. See registerItemDetailsProvider()[456] and ItemDetailsProvider[457] for details.

---

456 https://wiki.almworks.com/display/structure2gmaster/JavaScript+API+Functions#JavaScriptAPIFunctions-window.almworks.structure.api.registerItemDetailsProvider(itemType,ProviderClass)

457 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class

There is also an API usage sample that implements item details for Jira users. It can be found on API Usage Samples(see page 1152) page, see 'user-item-details'.

7.21.7 Structure 5.2.1 Release Notes

> **6th of February 2019**
>
> Structure 5.2.1 fixes a critical indexing problem that may affect users.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[458]

7.21.7.1 Patch Release

This is a patch release based on Structure 5.2.0. It fixes a critical problem that could cause a full re-index to fail because of interference from Structure-related background activity.

Upgrade is recommended for all customers using Structure 5.2.0.

7.21.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.21.7.3 Upgrade

> If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 5.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

458 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.21.7.4 Enterprise Deployment Notes

The main platform change in Structure 5.2 was the introduction of index consistency checking. To summarize, Jira 8.0 includes a newer version of Lucene, so it will delete your existing index after the upgrade and start re-indexing in the new format. Since Structure relies heavily on the Lucene index, we want to pause its operation until the re-index completes and the index becomes consistent with the database again.

To do that, Structure periodically checks the Lucene index consistency by calling a method provided by the Jira API, which performs a few database queries and index accesses. We have discussed our implementation with the Jira development team, who assured us that these periodic checks wouldn't negatively affect the performance of the Jira instance.

During a full re-index, the Lucene index is normally expected to be inconsistent, and the consistency checks should be suspended. However, due to a problem with the Jira API not reporting correctly that the index is being rebuilt, Structure was still performing these checks in the background once about every 5 seconds. In a few support cases, we have observed a strong correlation between this background activity and the re-index failing due to missing index files.

We are communicating with Jira developers to uncover the actual cause of the issue. In the meantime, to work around the problem, Structure 5.2.1 uses a different method to detect that a full re-index is in progress. Also, we have made it possible to disable index consistency checking altogether by setting a dark feature.

We advise that you try running a full re-index on a staging Jira instance before upgrading from Structure 5.1 or earlier. The usual load and stress testing can also be performed.

Need help or have questions? Contact Tempo Support[459].

7.21.8 Structure 5.2.2 Release Notes

1th of March 2019

Structure 5.2.2 fixes a regression and improves compatibility with Jira 8 and Portfolio 2.24.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[460]

7.21.8.1 Patch Release

This is a patch release based on Structure 5.2.1. It fixes a regression, introduced in version 5.2.0, which could manifest in the following:

- Generators and synchronizers can produce partial or incorrect results when background re-index is in progress.
- Synchronizers don't react to issue changes made by other synchronizers.

This version also improves compatibility with Portfolio 2.24 and Jira 8 and addresses the following issues:

- Fixed: Parent Issue Grouper does not work with Portfolio 2.24.

459 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

460 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

- Fixed: Issue fields with dropdowns are not editable from Gadget on Jira 8.
- Fixed: `NoSuchMethodException` when flushing thread-local searchers.

Upgrade is recommended for all customers using Structure 5.2.x, Jira 8, or Portfolio 2.24.

7.21.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.21.8.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure for versions 3.x to 5.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.21.8.4 4. Enterprise Deployment Notes

Structure 5.2.2 does not introduce changes that could affect performance or would justify additional load and stress testing for Enterprise deployments.

7.22 Structure 5.1 Release Notes

25th of October, 2018

Structure 5.1 adds wiki markup, an admin interface for dark features and several performance improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 1164) Try It: Structure Sandbox Server[461] (no installation required)

7.22.1 Version Highlights

- Formula columns now support wiki markup

461 https://demo-structure.almworks.com/

- New admin interface for enabling/disabling Structure dark features
- Inactive users are now flagged when grouping by Assignee
- Several performance improvements and fixes

⚠ Data Center customers: Due to a problem in the Atlassian Universal Plugin Manager, the previous Data Center approved version of Structure, Structure 5.0.1, was reported as incompatible. This problem does not occur with Structure 5.1.

7.22.2 Changes in Detail

7.22.2.1 Wiki Markup

Wiki markup is now supported within Formula columns(see page 813).

Using wiki markup, you can:

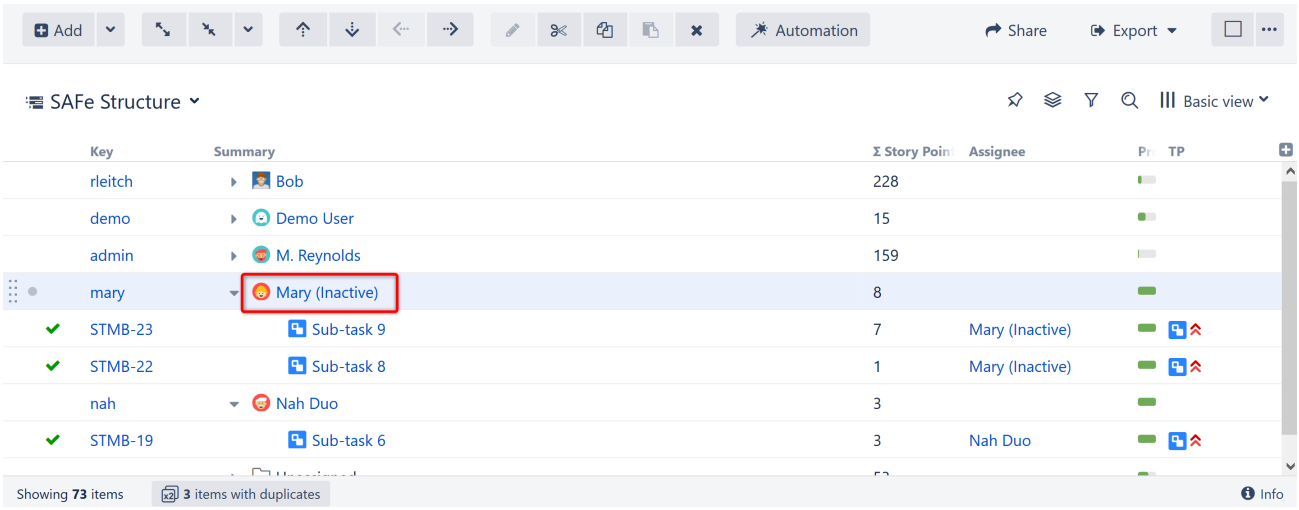
- Specify the text color within a column
- Highlight cells with background coloring
- Insert images
- Add emoticons

Markup content can be included when exporting a structure to Excel or a printable page.

To learn more, see Wiki Markup in Formula Columns(see page 251).

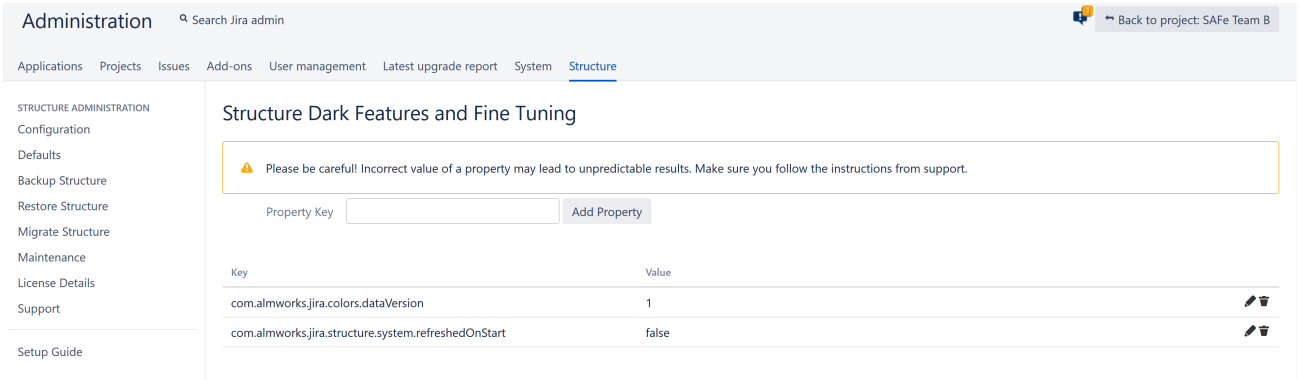
7.22.2.2 Inactive User Flag

When grouping by Assignee (either through automation or transformation), the "Inactive" identifier is now visible beside the group name:



7.22.2.3 Admin Interface for Dark Features

It is now possible to easily enable or disable dark features directly from a new Structure dark features interface.



To access the interface, you must have Jira Administration permissions and enter the interface location directly into your browser: https://YOUR_JIRA_ADDRESS/secure/admin/StructureDarkFeatures.jspa

Notes

- Properties added with our admin interface are added to Application properties
- A Structure property gets its value from Application properties; if no value is found, the property gets its value from System properties

For more information, see Advanced Configuration(see page 813).

7.22.2.4 Notable Fixes and Improvements

- Group, project and role lookup is now available when editing structure permissions
- The #distinct modifier has been added to the JOIN function in formulas, to prevent values from being added multiple times
- Shared perspectives can be viewed with anonymous access - issues that require specific permissions will not be shown in the structure
- When adding generators, the "allow changes" option can be unchecked by default
- Fixed: Exporting to Excel issue when exporting a perspective created by Structure 4.x

- Fixed: Unable to move issues under a folder when using Rank sort generator
- Fixed: Clicking column header does not sort structure when user has view permission
- Fixed: Reordering sub-tasks on the issue page does not reorder issues in the structure
- Several additional fixes and performance improvements

7.22.3 Supported Versions

Structure 5.1 and all extensions support Jira versions from 7.2 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.0. Structure.Testy[462] extension, Colors[463] , Structure.Pages[464], Structure.Gantt[465] and integrations with third-party apps should continue working normally.

> Structure 5.1 will be the last version to support Jira 7.2-7.5. Structure 5.2 will support Jira 7.6+.

7.22.4 Installation and Upgrade

7.22.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 1164) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on the Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please note that there are limitations to its compatibility with Structure 4.2 and higher, so a Confluence upgrade to version 6.1 or later is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.22.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure from versions 3.x–5.0 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.

---

462 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
463 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
464 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
465 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

> We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

7.22.5 Enterprise Deployment Notes

Structure 5.1 has a number of changes that are important for large-scale Jira Server and Jira Data Center instances.

7.22.5.1 Structure Index Monitor Field DATA CENTER

"Structure Index Monitor" is a system custom field that is automatically installed on Jira Data Center instances since Structure 4.6. It helps Structure capture events related to Lucene-based index subsystem. This is what it looks like on the Jira administrator's Custom Fields page:

This custom field does not store any data in the index, so it does not have a performance impact. Structure Index Monitor field cannot be removed, but it will disappear if Structure is uninstalled and will not affect Jira at all.

One application of this field is ensuring correct cache operation in Structure. When an issue is updated on one node, the indexes on all nodes must be updated to make sure JQL queries return the correct result everywhere. Structure caches also need to be checked on all nodes, and potentially a JQL query should be re-run to make sure that generated content is up to date. The cache update should not happen earlier than the index update, otherwise a JQL query would return an outdated result.

With Structure 5.1, we use the Structure Index Monitor field to help with the timing of cross-node updates. It eliminates a chance of a race condition that would cause Structure to show outdated content on a particular node for a long time after a relevant update has happened.

To test the rollout of this change, you can observe the behavior of the system under a constant stream of issue updates (one change per second, for example), happening on one node, while a user observes a structure with JQL-based automation on another node. At some point the stream of changes should stop, and a few seconds later the user should see the most up-to-date information in Structure.

This feature has been experimental for a while, and we're enabling it now as the default. It is still possible to turn it off and go back to the way events were propagated between nodes in Structure 5.0 and earlier, by using the dark features interface and setting `structure.delegatingItemTracker.enableReindexMonitor` to `false`.

7.22.5.2 Offloading Change Stream Writing to a Background Thread DATA CENTER EXPERIMENTAL

In the Data Center environment, Structure running on one node needs to let Structure running on other nodes know when an item (an issue or some other object) changes. This "change stream" is communicated to other nodes via the database, asynchronous caches and occasional use of a global, one-per-cluster lock.

Normally, each change is written into the database immediately when it happens – in the "event listener". That code runs in the same execution thread as the change itself, typically as a response to a user's action. We recently worked on an incident where the Jira global locking subsystem failed and there were certain issues with the

database. That made writing of the change stream "hang", which, in turn, made user request threads hang, which led to Jira being unresponsive.

In Structure 5.1, we're introducing an alternative implementation of this subsystem, which would never block a user request thread. All global lock operations and writing to the database will happen in a separate, dedicated thread of execution.

This feature is currently **not** enabled by default. For now we recommend to turn it on if a similar problem has happened in the past, or if there's a problem with Jira hanging and you suspect it might be this case.

To enable this feature, set the `structure.delegatingItemTracker.enableAsyncEventsQueue` property to `true` in the dark features interface. For more information, see Advanced Configuration(see page 813).

To test the change, use the same method as described in the previous section.

7.22.5.3 Lucene and Jira Search Integration

In preparations for the upcoming Jira 8 release, we have adjusted how Structure works with Lucene index and Jira search. This change should not affect performance or the users in any way. To test that everything is okay, one could run a JQL query from Structure | Query board.

7.22.5.4 Bulk Loading Rows from the Database

One of the database tables that Structure most frequently uses is `AO_8BAD1B_ROW`. Structures contain rows, and this table stores the mapping between unique numeric "row IDs" and arbitrary "item IDs" of the objects that structures contain. This table is used very often when a structure is displayed.

With Structure 5.1, we implemented a bulk operation that lets Structure get multiple row information with one database request. That greatly speeds up loading structures that have lots of manually added issues and reduces the database load.

To test this improvement, you can try creating a temporary structure, adding several thousands of issues there (manually, copying them from a search result), and then opening that structure after an upgrade or after Structure is disabled and enabled.

7.22.5.5 Testing on Staging Environment

Apart from the specific suggestions given above, you can run the generic load and stress testing on a staging environment as advised in the previous release notes.

Need help or have questions? Contact Tempo Support[466].

7.22.6 API Changes in Structure 5.1

7.22.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

466 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Jira Version	New API Version
7.2+	16.11.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.22.6.2 Compatible Changes in the Java API

- The new `DarkFeatures` class, backed by Jira's `ApplicationProperties`, is used to obtain advanced configuration parameters where previously Structure relied on system properties. It is primarily intended to be used by Structure itself and other ALM Works products.
- `La.cast()` is a new static factory method that produces a `La` instance that safely casts objects to the given class. It complements the existing `La.instanceOf()` factory method.
- A new version of `RowTree.appendForest()` takes a `java.util.function.LongPredicate` instead of the `LongPredicate` from HPPC. The old method has been deprecated.

7.22.7 Structure 5.1.1 Release Notes

5th of May 2020

Structure 5.1.1 is a security patch release.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[467]

7.22.7.1 Patch Release

This is a patch release based on Structure 5.1. We have addressed a security issue that affects all Structure versions starting with 3.5. Problem was fixed in the latest (6.0) Structure.

Upgrade is required for all Jira instances running Structure version 5.1 and using Jira 7.2.x.

You should have Structure license with active maintenance (expiring not earlier than May 6th, 2020) to upgrade.

7.22.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.

467 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.22.7.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes[468].

The upgrade procedure from versions 3.x–5.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.23 Structure 5.0 Release Notes

> **16th of August, 2018**
>
> Structure 5.0 introduces Horizontal Scrolling, Manual Adjustments to auto-generated content, user interface updates and more.

Download the latest version of Structure and its Extensions
(see page 1164)Try It: Structure Sandbox Server[469] (no installation required)

7.23.1 Version Highlights

- Manual adjustments to auto-generated content
- Horizontal scrolling
- Work Logged can now be based on user, group or role
- Flexible maintenance scheduling with crontab
- Support for “hidden” issue links
- User interface changes to match Jira 7.10

7.23.2 Changes in Detail

7.23.2.1 Manual Adjustments

Manual Adjustments allow you to move dynamic content within a structure, regardless of the Automation used to create the structure.

When you use generators to build structures, you’re restricted in how you can move items within your structure, based on the generator rules. When Manual Adjustments are enabled, you can move items or create new items within your structure regardless of the generators used to create it.

468 https://wiki.almworks.com/display/structure2gmaster/Structure+3.0+Release+Notes
469 https://demo-structure.almworks.com/

Manual Adjustments are not reflected in Jira and may be affected by changes to your generators or Jira items.

Documentation: Manual Adjustments (see page 194)

Implementing Manual Adjustments required us to make changes to our backup file format. As such, previous versions of Structure will not be able to restore data from backup files created by Structure 5.0 and later. We strongly recommend that you back up your data before upgrading. For more information, see Backup Format Version Change(see page 823).

7.23.2.2 Horizontal Scrolling

To make it easier to work with multiple columns and/or larger column widths, you can now enable horizontal scrolling within your structure. When Horizontal Scrolling is enabled, you can add columns beyond the width of your viewing pane and navigate between columns using a convenient horizontal scrollbar.

Documentation: Horizontal Scrolling(see page 503)

7.23.2.3 Work Logged by User, Group or Role

The Work Logged column has been updated so time spent on an issue can now be calculated for a specific user, group, or project role.

Documentation: Work Logged Column(see page 499)

7.23.2.4 Automatic Maintenance Scheduling with Crontab

Automatic Structure Maintenance now supports flexible scheduling with crontab.

Using standard crontab formatting, you can customize your maintenance schedule to request that tasks run only on certain days or during certain times, process multiple backups each day, specify exact times down to the minute and more.

Documentation: Automatic Structure Maintenance(see page 966)

7.23.2.5 Other Changes

Other changes include:

- Structure now supports hidden issue links
- Several user interface changes were made to match Jira 7.10 graphical changes and improve accessibility

7.23.3 Supported Versions

Structure 5.0 and all extensions support Jira versions from 7.2 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.6. Structure.Testy[470] extension, Colors[471], Structure.Pages[472], Structure.Gantt[473] and integrations with third-party apps should continue working normally.

7.23.4 Installation and Upgrade

7.23.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from Atlassian Marketplace or from Download(see page 1164) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.23.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x–4.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

> We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 823).

7.23.5 Enterprise Deployment Notes

Structure 5.0 introduces the ability to manually adjust generated content and improves the performance of the Sub-Task extender. The changes in the Automation sub-system can be particularly important for larger installations and Jira Data Center instances.

---

470 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
471 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
472 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
473 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

7.23.5.1 Controlling Manual Adjustments

Manual adjustments are disabled by default for new and existing structures, so they shouldn't immediately affect Structure's behavior and performance.

Structure's permission system is used to control who has access to manual adjustments:

- Enabling manual adjustments for a structure requires **Control** permission for that structure.
- Any user with the permission to create structures can create a new one and enable manual adjustments for it.
- **Edit** permission is required to add and modify manual adjustments if they are enabled.
- **Control** permission is required to remove all manual adjustments for a structure.

Documentation: Structure Permissions(see page 543), Changing Permission to Create New Structures(see page 955).

In addition to permissions, there are configurable limits on the number of manual adjustments:

- 200 adjustments per user action, controlled by the system property `structure.gfs.manualAdjustments.maxAdjustmentsPerAction`
- 2000 adjustments per structure, controlled by the system property `structure.gfs.manualAdjustments.maxAdjustmentsPerStructure`

Finally, manual adjustments can be disabled for the entire Jira instance by setting the `structure.gfs.manualAdjustments.enable` system property to `false`.

Documentation: Advanced Configuration with System Properties(see page 820)

7.23.5.2 New Database Table for Manual Adjustments

We have added a new table named `AO_8BAD1B_MANUAL_ADJUSTMENTS` to the database schema. This table stores primary data (not stored anywhere else in Jira).

The table contains one row for each structure with manual adjustments. The total amount of data stored depends on the number of adjustments and the complexity of the structure, however, a rough estimate would be on the order of 100 characters per adjustment.

The data from this table is exported along with other Structure data when you do Structure Backup in a human- and machine-readable form. The table is also exported along with all Structure data during Jira backup.

7.23.5.3 Backup Format Version Change

In Structure 5.0 we had to change the backup file format version due to the backwards-incompatible changes required by the introduction of manual adjustments. This means that previous versions of Structure will not be able to restore data from backup files created by Structure 5.0 and later. Because of this, we strongly recommend that you back up your data before upgrading.

However, if you want to downgrade to an earlier version, you can use the procedure outlined below to restore Structure data from a 5.0 backup file:

- Unpack the XML backup file from the ZIP archive created by Structure.
- Change the `version` attribute in the `<structure-backup>` element from "5.0" to "3.3".
- Delete all `<manualAdjustments>` elements from the XML.

Then you can restore directly from the modified XML file; you do not have to pack it into a ZIP archive.

7.23.5.4 Sub-Task Extender Performance Improvement

The Sub-Task extender has to check whether each issue in its scope is a sub-task or a regular issue. Previous versions of Structure relied on JQL to perform issue type checks, and on large Jira instances with hundreds of issue types these JQL queries caused significant performance impact. In Structure 5.0 we have changed the Sub-Task extender to read issue types from the Lucene index directly, instead of using JQL. We have also improved caching, so sub-task extenders are now better able to reuse the results of issue type checks done by other sub-task extenders.

7.23.5.5 Testing on Staging Environment

The introduction of manual adjustments required extensive changes throughout the Automation sub-system, so we advise you to perform load and stress testing in a staging environment before you upgrade. You can try the following scenarios:

- If you have large structures (10,000 issues or more) built by generators, especially sub-task extenders, try opening those from a number of client computers using different user accounts.
- Make some changes to Jira issues that would cause these structures to regenerate – change field values, add or remove links, change sub-task parents.
- Make changes in the structures that would be processed by generators – move issues to different groups, move linked issues and sub-tasks to different parent issues.
- Create a new structure and add manual adjustments to it. For example, you can rearrange a flat list of issues added by a JQL inserter into a hierarchy, or move the issues into several manually added folders.

Watch the log files for errors and warnings while running these experiments.

Need help or have questions? Contact Tempo Support[474].

7.23.6 API Changes in Structure 5.0

7.23.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, except those in `@Internal` interfaces.

Jira Version	New API Version
7.2+	16.10.0

See Structure API Versions(see page 1069) for full version information and downloads.

474 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.23.6.2 Compatible Changes in the Java API

Horizontal scrolling

We have added a new parameter named "column display mode" to the view specification, with three new getters and setters:

- `ViewSpecification.getColumnDisplayMode()`
- `ViewSpecification.Builder.getColumnDisplayMode()`
- `ViewSpecification.Builder.setColumnDisplayMode(int)`

Supported values are stored as `int` constants in the new class `ColumnDisplayMode`:

- `AUTO_FIT` means that the columns fill the entire width of the panel, growing or shrinking as necessary;
- `SCROLLABLE` means that the columns always have their manual or preferred widths, and a horizontal scroll bar appears if needed.

The default column display mode is `AUTO_FIT`.

New method in the `Forest` interface

The new method `copySubforestAtIndex(int)` returns a new forest that contains all descendants of the row at the given index. It is a counterpart to the existing method `copySubforest(long)`, and it allows you to save an index lookup when you already have the index. The method is implemented in `ArrayForest`.

New row semantics

We have added two new codes to the `CoreSemantics` interface:

- `INSERTED` marks rows added by inserter generators;
- `EXTENSION` marks rows added by extenders.

Please note that row semantics is still a work in progress, and `CoreSemantics` is marked `@Internal`.

Miscellaneous fixes

A few minor changes in `AbstractAggregateLoader`, `AbstractDistinctAggregateLoader`, and `ReductionStrategy` are meant to prevent `NullPointerException` in certain circumstances.

We have also added reentrancy protection to the item lookup code in `StructureException`. Please note that item lookup is disabled by default since version 16.7.0 (Structure 4.4).

7.23.6.3 Incompatible changes in `@Internal` interfaces

The method `ForestAccessCache.removeOutOfScopeIssues()` was moved to a separate interface, `ProjectScopeCache`. We have also added a new internal interface named `SubTaskIssueCache`. Neither of the three interfaces is intended to be used by third-party developers at this point.

7.24 Structure 4.6 Release Notes

27th of March, 2018

Structure 4.6 introduces Automation Timeout safeguard and inline editing of the Status field.

Download the latest version of Structure and its Extensions
(see page 1164)Try It: Structure Sandbox Server[475] (no installation required)

7.24.1 Version Highlights

- Automation Timeout safeguards Jira against excessive load that may result from misconfiguration of Structure generators
- Inline editing of the Status field enables users to change an issue's status with a double-click on the Structure grid

7.24.2 Changes in Detail

7.24.2.1 Automation Timeout

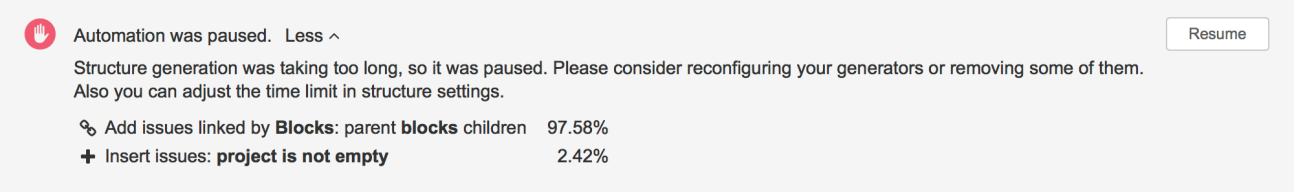
The Structure Generators(see page 140) feature is very powerful, and very flexible. It gives the user the building blocks – generators – for constructing dynamic structures. As with any powerful and flexible feature, it's possible to unintentionally create an unanticipated load on the Jira server. The Automation Timeout is a safeguard that protects Jira from unintended server loads.

Some generators already have a considerable level of protection. For example, JQL Inserter offers an option to limit the maximum number of issues it adds to a structure. Extenders have a maximum number of levels to expand a hierarchy. Automation Timeout adds universal protection on top of that by limiting the amount of time Structure app can spend generating a particular structure.

Here's how it works.

When Structure app detects that a structure is being generated for an unreasonably long time (by default, 30 seconds), it **pauses Automation** for that structure. The Structure panel will show the "**Automation was paused**" message and the structure itself will miss the generated content. If the structure contains only dynamic elements, it will appear empty.

If the user clicks "More" link, they can inspect which generators took most of the time.



Any user who can view the structure can resume Automation by clicking **Resume** button. This will cause Structure to try and generate this structure again.

---

475 https://demo-structure.almworks.com/

Note that sometimes structure generation may time out because of general slowness of Jira caused by other factors. It is ok to click Resume and try again in that case, without changing the generators.

Documentation: Paused Automation(see page 826)

7.24.2.2 Inline Status Editing

It is now possible to edit an issue's Status inline by double-clicking on its current status in the Status column.

You can then select a new status from a list. The list will show only the statuses that could be the issue's next status, according to the workflow. If a particular transition includes a screen with additional fields, this transition cannot be executed through inline editing and will require the use of Jira Actions(see page 139).

Documentation: Editing Issues from Within Structure(see page 132)

7.24.2.3 Notable Fixes and Improvements

- Fixed: When exporting a structure to a Printable page in Automation Editing Mode, some rows are not shown
- Fixed: When exporting a structure to Excel, large texts are truncated to 16382 symbols
- Fixed: Issues can be added to archived Fix Versions
- Fixed: Progress based on Time Tracking takes folders and pages into calculation
- Fixed: Folders are not shown in the structure on a product page
- Fixed: Duplicates filter is not included in a shared perspective

7.24.3 Supported Versions

Structure 4.6 and all extensions support Jira versions from 7.2 to 7.8. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.5. Structure.Testy[476] extension, Colors[477], Structure.Gantt[478] and integrations with third-party apps should continue working normally.

7.24.4 Installation and Upgrade

7.24.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from Atlassian Marketplace or from Download(see page 1164) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please, note that there are limitations to

476 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
477 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
478 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.24.4.2 Upgrading Structure

⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x–4.5 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.24.5 Enterprise Deployment Notes

Structure 4.6 has a number of changes that are particularly important for large installations and Jira Data Center instances.

7.24.5.1 Automation Timeout

One of the problems we've seen in the past is that one or more improperly configured structures would take a lot of time to generate, consuming system resources. In extreme and rare occasions, system load could make entire Jira instance unresponsive. To prevent that from happening, we're introducing a time limit for structure generation. If a structure takes longer time to get generated, it will be marked as "paused", and all generators in it will be temporarily disabled.

The default timeout is 30 seconds. If some of your structures currently take a long time to generate, they might get paused after the upgrade.

Therefore, we advise you to carfeully check your all of your bigger and mission-critical structures in a staging environment before upgrading and adjust the timeouts as necessary. The structures with paused automation will be marked as such on the Manage Structures page.

✓ Note that someone needs to actually open a structure for Automation to run the generation process. A structure that is not being looked at does not consume any system resources.

The timeouts can be adjusted for each structure using "Configure" link on the Manage Structure page. The global default can be adjusted in the **Administration | Structure | Defaults** menu section.

7.24.5.2 Synchronizers on Jira Data Center

Structure uses only one cluster node to run synchronizers. We've seen some problems with the mechanism used to select a cluster node for running synchronization and hand off its work to a different node in case the original node is stopped or loses connectivity. These problems could cause all synchronizers in the system to stop working.

In this version we're introducing a simpler and more reliable mechanism which uses conditional database updates and Jira's built-in node aliveness checks. A new database table, AO_8BAD1B_ATOMIC_FLAG, is introduced to support the new solution. In this version, it is expected to contain only a single row that shows which node is currently responsible for running synchronizers.

You can use the following steps to test syncronization node selection and hand-off:

1. Start Jira (Data Center), install Structure 4.6 and check application logs for messages from *ClusterExclusiveWorkNodeFlag* to determine which node is currently responsible for synchronization.
2. Shut down or disconnect that node and wait for up to 6 minutes, which is the maximum reaction time.
3. Check application logs for messages from *ClusterExclusiveWorkNodeFlag* to see that synchronization was taken over by another live node.

7.24.5.3 Lucene Index Monitoring

Structure relies on the Lucene index to check users' access to issues, or quickly read a single value for a large number of issues. The index may be temporarily inconsistent during full or project reindexing, index replication or recovery. Structure 4.2 introduced re-index monitoring as a part of a dark feature used to work around a race condition in Jira Data Center. We've updated this mechanism in Structure 4.6 to also handle index replication and automatic recovery, and now we're enabling it for all instances, so that Structure is able to recover when it detects massive index changes.

After the upgrade, a new, undeletable system custom field called "Structure Index Monitor" should appear. This field is used only to track the "reindex" events. It does not contain any data and will not have any effect on issues. If Structure is uninstalled, this custom field disappears.

7.24.5.4 Old Row Manager Implementation Removed

Structure 3.4 introduced a new implementation of one of its core components used to store temporary rows in the generated structures. The old implementation remained in the code with the ability to switch to it by setting a system property. In over a year since that release the new implementation has proven to be reliable, and we never advised our customers to switch to the old one. In Structure 4.6 we remove the old implementation and its dependencies completely.

If you see folders named *rows0*, *rows1*, etc., under *JIRA_HOME/structure* folder, they can be safely deleted – they were a part of the older implementation.

7.24.5.5 Testing on Staging Environment

There are no particular special areas of interest for load testing and stress testing Structure 4.6. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[479].

479 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.24.6 API Changes in Structure 4.6

7.24.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.2+	16.9.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.24.6.2 Compatible Changes in the Java API

Automation timeout

We have added two new methods to notify Structure API users that the generation of a structure was stopped due to a timeout:

- `ForestSource.getHealth()`
- `VersionedForestUpdate.getHealth()`

These methods return a `ForestSourceHealthStatus` object, and its `isStopped()` method returns true if generation was stopped.

`StructureErrors.AUTOMATION_FAILED` is a new error which may be thrown when automation is paused and there is no way to return a health status object.

We have also added a new method named `isStopped()` to the `GenerationContext` interface. If you implement your own generators, we advise you to check this method before potentially long-running operations and inside loops, and return control back to Structure if it returns true.

Convenience methods in `StructurePropertyService`

We have added two new methods to the StructurePropertyService interface:

- `getLong(long, String)` returns the property value as a `long`;
- `setValue(long, String, long)` sets the property to the given `long` value.

`@Internal` components and classes

Two interfaces have been moved from Structure core to a new API package, `com.almworks.jira.structure.api.statistics`, for consumption by other ALM Works products. They are not intended to be used by third-party developers at this point.

7.24.7 Structure 4.6.1 Release Notes

27th of April 2018

Structure 4.6.1 fixes a compatibility issue with Jira Service Desk and a couple of compatibility issues with other apps.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[480]

7.24.7.1 Patch Release

This is a patch release based on Structure 4.6.

Upgrade is recommended for all customers who use Structure with Jira Service Desk.

We have addressed the following issues:

- Fixed: Jira Service Desk users without access to Structure can't create and edit queues.
- Fixed: Epic Sum Up App time field in a structure view updates unexpectedly.
- Fixed: Xray App execution status is not rendered correctly in Structure column.

You should have Structure license with active maintenance (expiring not earlier than April 27th, 2018) to upgrade.

7.24.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.24.7.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)

480 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.24.7.4 Enterprise Deployment Notes

Structure 4.6.1 does not introduce changes that could affect performance or would justify additional testing for Enterprise deployments.

7.24.8 Structure 4.6.3 Release Notes

> **11th of May 2018**
>
> Structure 4.6.3 fixes a problem in Performance Audit Log diagnostics system and has several performance improvements.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[481]

7.24.8.1 Patch Release

This is a patch release based on Structure 4.6.1. (There was no 4.6.2 release.)

We have fixed a problem that caused collection of Performance Audit Log to fail when Jira had a saved filter with an empty JQL, a JQL with only an "order by" clause, or an invalid JQL. This problem could prevent collection of valuable diagnostic data for ALM Works support team.

We also added five various performance improvements that are important for large instances (Server and Data Center), especially when there are frequent issue changes (for example, one issue per second).

Upgrade is recommended if you have encountered the problem with Performance Audit Log or if you have a large Jira instance.

> You need a Structure license with active maintenance (expiring not earlier than May 11th, 2018) to upgrade.

7.24.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from the Download(see page 1164) page.
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

481 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.24.8.3 Upgrade

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.24.8.4 Enterprise Deployment Notes

There is a number of small changes in Structure 4.6.3 that are not visible to the user (except for improved performance).

The following a is a list of affected components with suggestions of how they could be tested on a staging server.

1. Attribute loading subsystem: it is used when values are loaded into the Structure grid. We removed additional checks for concurrent changes that happened during the loading of attributes. This will speed up the attribute loading, and on a busy system it will drastically increase the responsiveness of the application. To test the subsystem, open different structures and make changes to the issues. Check how totals and formulas are being recalculated. Try to load-test the staging instance by having a constant flow of issues changes while a structure with an inserter and a generator are being used.
2. Sub-task extender, Agile Rank sorter, Agile synchronizer, Sub-task synchronizer: the performance of these components was improved by changing the way issues are checked for being sub-tasks. The performance improvement will have bigger effect on systems with hundreds of issue types.
3. Automation subsystem: eliminated some inefficiencies during generator execution. To test, check the most frequently used automated structures. Try changing values in Jira and see the generated structure update. If you are using issue security, verify a limited user's access to issues. Check how transformations and quick transformations work.

Additionally, we improved the user interface of Performance Audit Log.

7.24.9 Structure 4.6.5 Release Notes

> ⓘ **13th of June 2018**
>
> Structure 4.6.5 adds compatibility with Jira 7.10 and several performance improvements.

Download from Archive[482]
Structure on the Atlassian Marketplace[483]

482 https://wiki.almworks.com/display/structure/Download+Archive
483 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.24.9.1 1. Patch Release

This is a patch release based on Structure 4.6.3. (There was no 4.6.4 release.)

Starting with this version, Structure is compatible with Jira 7.10.

There are two known issues:

- We are still working on making Structure's user interface match the new look and feel of Jira 7.10. The app is perfectly usable, but you can still see some older-style controls. We are planning to complete the redesign in the next feature version.
- Archiving a project, which is now possible with Jira 7.10 Data Center edition, will cause all archived issues to disappear from all structures. See Archived Projects and Structure(see page 971) for details.

This upgrade is recommended for all large instances. Depending on Structure usage, this release may greatly improve Structure performance – specifically, reducing the time it takes to load structures with automation based on Portfolio or Jira links. The improvement is especially noticeable on larger instances with a large number of projects.

This upgrade is required for instances running Jira 7.10.

> ⚠ You need a Structure license with active maintenance (expiring no earlier than June 13th, 2018) to upgrade.

2. Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from the Download[484] page .
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

3. Upgrade

If you are upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

The upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up Jira data, using **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

4. Enterprise Deployment Notes

Structure 4.6.5 has a few performance improvements especially important for large Jira instances with hundreds and thousands of projects.

484 https://wiki.almworks.com/display/structure/Download

When Structure is enabled only for certain Jira projects, generators have to make sure that they are adding issues only from those projects. Structure relied on JQL to perform project checks, producing JQL queries with clauses like: `project IN (<list of selected project IDs>)`. On large Jira instances with thousands of projects and hundreds of those projects using Structure, these JQL queries cause significant performance impact, especially when run very often by Portfolio, Link, and Agile extenders.

In Structure 4.6.5, we have changed the project checking code to read project IDs from Lucene index directly, instead of using JQL. We have also improved caching, so generators are now better able to reuse the results of project checks done by other generators. We have seen significant performance improvements with these changes in the scenario described above.

The changes affect Portfolio, Link, and Agile extenders and, to a lesser extent, JQL, Agile, and Text Query inserters. If you have large structures (about 10,000 issues or more) built using these generators, we advise you to perform load and stress testing on a staging environment before upgrading, even if your Jira instance doesn't have a lot of projects.

Need help or have questions? Contact Tempo Support[485].

7.25 Structure 4.5 Release Notes

28th of December, 2017

This version introduces duplicate items highlighting, which makes it much easier to find, review and delete duplicate issues in a structure. This version also includes several performance improvements and bug fixes.

Download the latest Structure and Extensions
(see page 1164) Structure Demo Server[486]

7.25.1 Version Highlights

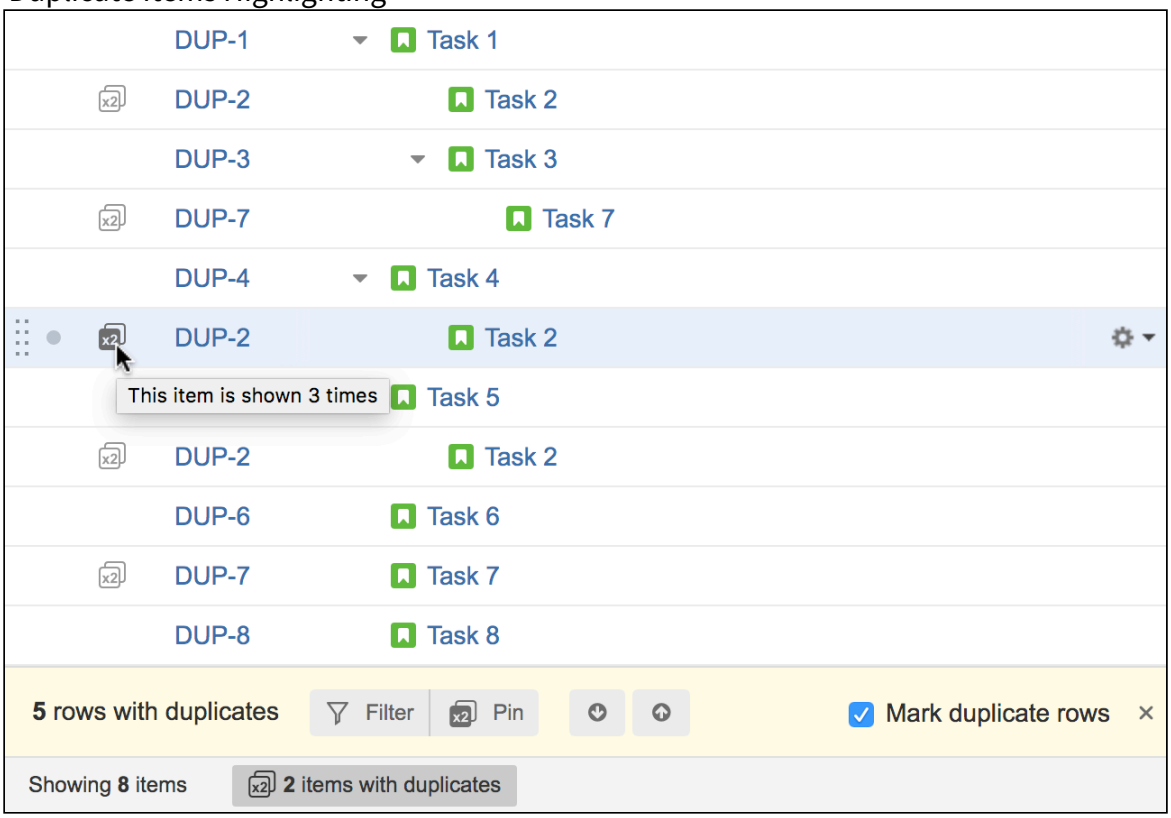
- Duplicate items highlighting
- Several performance improvements and fixes

485 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
486 https://demo-structure.almworks.com/

7.25.2 Changes in Detail

7.25.2.1 Duplicate Items Highlighting

Issues can appear in



structure more than once both when added manually or automatically. You can now quickly find, highlight and switch between such issues in a structure. This makes working with duplicates much simpler. For example, if necessary, you can delete manually added duplicates or adjust automation settings to make sure automation only adds one instance of an issue.

Documentation: Identifying Duplicate Items(see page 210)

7.25.2.2 Notable Fixes and Improvements

- When the issue details panel is closed, previous layout is restored
- Fixed: Performance Audit Log throws an error if Jira saved filters are used in generators
- Fixed: Level option is not saved when quick transformations are created
- Fixed: Unresolved issues from closed sprints do not show under Backlog when grouped by sprint
- Fixed: Browser search prompt changes drag and drop behavior in Structure to Copy when Move is expected
- Fixed: Show Results button does not appear when backing up Structure in a Jira Data Center instance
- Fixed: Text Search inserter moves modified issues to the bottom
- Various performance improvements

7.25.3 Supported Versions

Structure 4.5 and all extensions support Jira versions from 7.2 to 7.6. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.4. Structure.Testy[487] extension, Colors[488] plugin and integrations with third-party add-ons should work.

7.25.4 Installation and Upgrade

7.25.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for .Pages and Helper add-on, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.25.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x–4.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.25.5 Enterprise Deployment Notes

Structure 4.5 has a number of changes that are particularly important for large installations and Jira Data Center instances.

487 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
488 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.25.5.1 Extender Performance Optimizations

Big and deep structures built with multiple extenders have been a source of performance problems for some of our customers. Structure 4.5 contains numerous optimizations aimed at reducing both the generation time and memory consumption for such structures.

The changes to the extender implementations and the automation engine are quite substantial, so if you rely on large structures (10,000 issues or more) built with extenders, we advise you to perform load and stress testing on a staging environment before upgrading.

7.25.5.2 Other Changes

A fix in the Sprint grouper required it to load more data from Lucene. We do not expect this to be a problem, but if you rely on the Sprint grouper, it might make sense to test it before upgrading.

We have also optimized the attribute sub-system to speed up aggregate calculations on large, deep structures.

7.25.5.3 Testing on Staging Environment

Given the changes described above, you can test the following scenarios:

- Create a large, deep structure built with one or more extenders, for example, 10,000 issues or more, organized into 5 or more levels of hierarchy. Change one or more issues in the structure — add or remove issue links, change sub-task parents, etc., depending on the extenders you are using.
- While the structure described above is open, add one or more aggregate columns to your view (e.g. “Count Leaves” or a sum of a numeric field).
- Create a large structure (10,000 issues or more) consisting of a JQL inserter and a Sprint grouper (make sure most of the issues in the structure belong to one or more sprints). Move one or more issues to a different sprint.

Watch the log files for errors and warnings while running these experiments.

The usual load and stress testing can also be applied.

Need help or have questions? Contact Tempo Support[489].

7.25.6 API Changes in Structure 4.5

7.25.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

489 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Jira Version	New API Version
7.2+	16.8.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.25.6.2 Compatible Changes in the Java API

Generic items

Generic items are simple items managed by Structure. They are much like Structure's folders, but they can also contain an icon and a description. Generic items are meant to be used by other ALM Works products and third-party apps that integrate with Structure. Server-side APIs for generic items were introduced in Structure 4.4, and with the current release we are shipping client-side and UI code, making it possible to use generic item types in third-party apps.

Documentation: Declaring a New Generic Item Type(see page 1066)

`GenerationContext` changes

We have added a few methods to the `GenerationContext` interface.

- `itemsUpdate(DataVersion)` returns an item version update since the given version. It can be used to invalidate item value caches kept in the generated forest.
- `putTempObject()` and `getTempObject()` allow the developer to associate arbitrary data with the current generation context. Unlike `putObject()`, `putTempObject()` does not keep the passed objects in memory after the current generation task completes.

`ArrayForest` changes

A new method named `addForestMutuallyExclusive()` was added to `ArrayForest`. It is a faster alternative to `addForest()` and `mergeForest()` that can be used when the added forest is guaranteed to be mutually exclusive with the forest it's added to.

We have also optimized the implementation of another similar method, `replaceSubtreesMutuallyExclusive()`, which now performs fewer memory allocations when called frequently.

Miscellaneous

- Added `StructureUtil.nonBlank()`, which is used throughout the code to protect against null, empty and blank strings.
- Added a copying constructor to `ProcessDisplayParameters`.

7.25.7 Structure 4.5.1 Release Notes

19th of January 2018

Structure 4.5.1 adds compatibility with Portfolio version 2.11 and fixes other small issues.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[490]

7.25.7.1 Patch Release

This is a patch release based on Structure 4.5 and providing compatibility with Portfolio for JIRA version 2.11.

Upgrade is recommended for all customers who use Structure with Portfolio 2.11.

> ⚠ You should have Structure license with active maintenance (expiring not earlier than January 19th, 2018) to upgrade.

7.25.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.25.7.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
> - If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.25.7.4 Enterprise Deployment Notes

Structure 4.5.1 does not introduce changes that could affect performance or would justify additional testing for Enterprise deployments.

490 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.26 Structure 4.4 Release Notes

29th of November, 2017

Now you can quickly change issue status simply by dragging it from one Status group folder into another. This version also contains several bug-fixes and some improvements for our performance diagnostics tool.

Download the latest Structure and Extensions
(see page 1164) Structure Demo Server[491]

7.26.1 Version Highlights

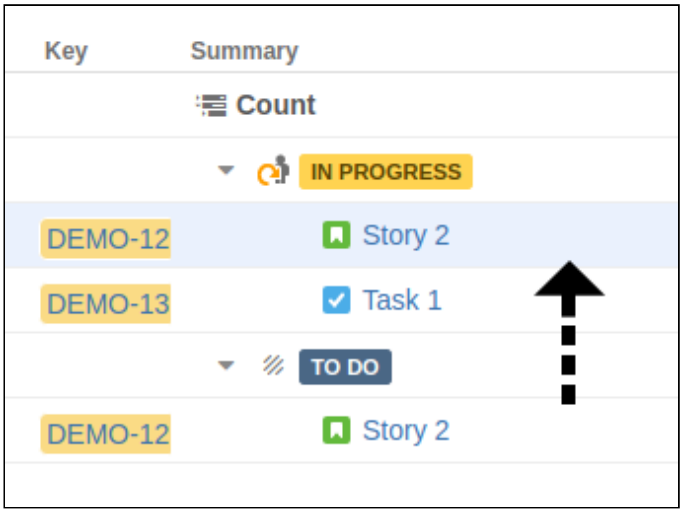
- Changing issue status using drag and drop.
- Support for Jira 7.6.
- Issue count is now displayed for the selected panel in double grid layout.
- Performance Audit Log includes more information.

7.26.2 Changes in Detail

7.26.2.1 Changing issue status in a structure

It is now possible to transition issues between statuses within a structure. Transitions that are allowed and do not have special workflow requirements (like required comments) can be done by simply moving issues between status groups.

Documentation: Groups(see page 169)



7.26.2.2 Totals in double grid

In the double-grid layout, the number of items is now displayed for the panel that is currently in focus, as opposed to only left panel.

491 https://demo-structure.almworks.com/

7.26.2.3 Changes to Performance Audit Log

Information about saved filters and filters used in generators or synchronizers is now available in Performance Audit Log.

It is now possible to save Performance Audit Log without scrolling to the bottom of the page: Save to File button is duplicated at the top.

7.26.2.4 Notable fixes and improvements

- Grouping by Customer Request Type is now possible with Service Desk 3.9.0.
- Fixed: Formula format would not work when *Sum over sub-items* checkbox was active.
- Fixed: Due In column would not show *Overdue* when past Due Date.
- Fixed: Work logged would not be displayed when filtered by users with usernames not in lower case.

7.26.3 Supported Versions

Structure 4.4 and all extensions support Jira versions from 7.2 to 7.6. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.3. Structure.Testy[492] extension, Colors[493] plugin, integrations with our partner add-ons should work with the new version.

7.26.4 Installation and Upgrade

7.26.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for .Pages and Helper add-on, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.26.4.2 Upgrading Structure

⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

---

492 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
493 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

Upgrade procedure from versions 3.x–4.3 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.26.5 Enterprise Deployment Notes

Structure 4.4 does not have a lot of core changes compared to Structure 4.3.

7.26.5.1 Index Monitoring Improvements

In Structure 4.2 we introduced an experimental "dark feature" for Jira Data Center to work around a race between Structure's issue change notifications and Jira's own inter-node messages which cause changed issues to be re-indexed on all nodes. You can refer to Structure 4.2 release notes, section 5.1(see page 853) for more information and detailed instructions.

Structure 4.4 improves this experimental event distribution subsystem to better handle full and single-project re-indexing operations. The changes are quite significant, so if you rely on this feature, it makes sense to test the new version on a staging environment before upgrading.

7.26.5.2 New Table for Generic Items

Generic items are simple items managed by Structure. They are much like Structure's folders, but they can also contain an icon and a description. Generic items are meant to be used by other Tempo products and third-party apps that integrate with Structure. Only the server-side APIs and components are released with Structure 4.4, so it is not possible for users to create generic items at this point.

With the introduction of generic items we have added `AO_8BAD1B_GENERIC_ITEM` table to the database schema. This table stores primary data (not stored anywhere else in Jira).

The data from this table is exported along other structure data when you do Structure Backup in a human and machine readable form. The table is also exported along with all Structure data during Jira backup.

We don't expect considerable database load on this table.

7.26.5.3 Testing on Staging Environment

There are no particular special areas of interest for load testing and stress testing Structure 4.4. We advise running the same testing procedures as you've done for previous upgrades.

Need help or have questions? Contact Tempo Support[494].

494 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.26.6 API Changes in Structure 4.4

7.26.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.2+	16.7.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.26.6.2 Compatible Changes in the Java API

Generic items

Generic items are simple items managed by Structure. They are much like Structure's folders, but they can also contain an icon and a description. Generic items are meant to be used by other ALM Works products and third-party apps that integrate with Structure. We are still working on this feature, and only the server-side APIs are released with Structure 4.4. Javadocs and developer documentation for generic items will be released in a future version.

`StructureException` changes

`StructureException` is thrown by many Structure API calls. In previous versions, if an exception was associated with an item or a row, it would add the item description to its message. This has proven problematic when fetching the item itself would fail, so in version 16.7.0 this feature has been disabled by default. You can enable it by setting the "structure.exception.enable.lookup" system property to "true", but this is not recommended on a production system.

`@Internal` components and classes

Several classes have been moved from Structure core to the API for consumption by other ALM Works products. Two new packages have been added – `com.almworks.jira.structure.api.effect` and `com.almworks.jira.structure.api.rest.effect`. They are not intended to be used by third-party developers at this point.

We have also added a new internal utility method, `StructureUtil.isDevMode()`.

7.27 Structure 4.3 Release Notes

20th of October, 2017

Structure 4.3 adds more aggregate functions for the Formulas columns, introduces support for the notes column in formulas and introduces filtering by Sprint status. This version also contains a minor security fix for those who use Notes column, other improvements and fixes.

Download the latest Structure and Extensions
(see page 1164) Structure Demo Server[495]

7.27.1 Version Highlights

- New aggregate functions.
- Modifiers of aggregate functions.
- Show and filter by Sprint statuses (future, closed, current).
- Notes column can now be used in formulas.

7.27.2 Changes in Detail

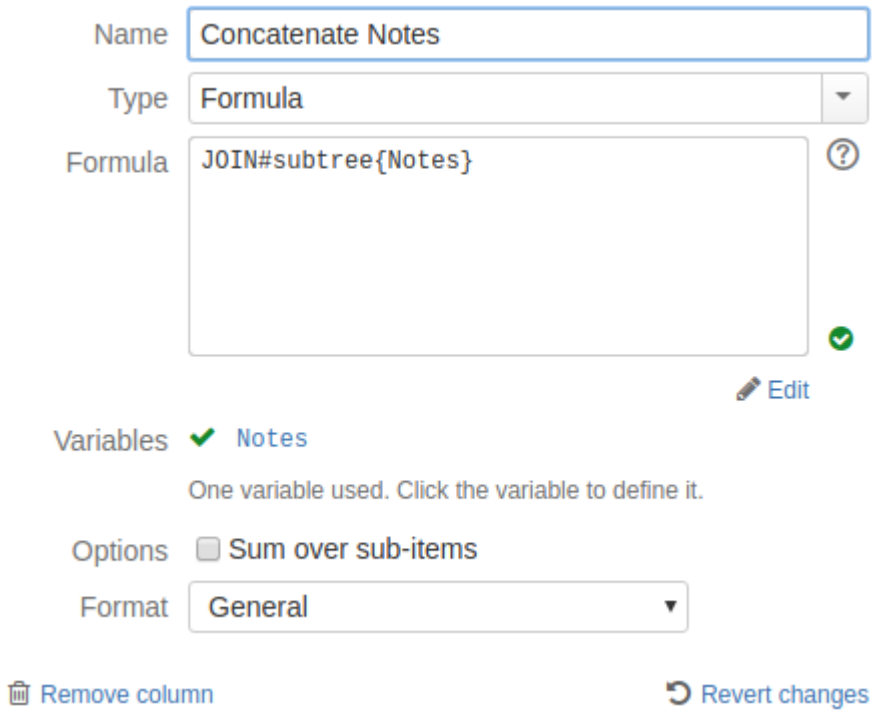
7.27.2.1 New Aggregate functions and modifiers

Further expanding on **aggregate functions** functionality, new functions are introduced: `COUNT{}`, `AVG{}`, `MAX{}`, `MIN{}`, `JOIN{}`, `PARENT{}`

Aggregate functions can also be modified to aggregate over different groups. Modifiers include but are not limited to *#all, #children, #subtree, #leaves*

Example of usage of the new syntax would be `JOIN#leaves{X}`, which concatenates contents of field X for all the leaves of the current item's sub-branch.

Documentation: Expr Language(see page 235), Aggregate Function Reference(see page 387)



495 https://demo-structure.almworks.com/

7.27.2.2 Using notes column in formulas

Notes column can now be used in formula column calculations as a variable.

It is important to mention, that notes column values are defined per structure, thus calculations including notes can return different results for the same issue in different structures.

7.27.2.3 Filtering by sprint status

Using filtering generator it is now possible to display only issues assigned to Future, Closed or Current sprints or combination thereof.

Documentation: Transformations page[496] and Filter page[497].

7.27.2.4 Security patch

We have addressed a minor security issue, affecting all Structure versions starting with 4.1 for those who use Notes column.

Details of the issue are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

If you cannot upgrade to Structure 4.3 because you are using Jira 7.1, please make sure to install Structure 4.1.1(see page 861).

7.27.2.5 Notable fixes and improvements

- It is now possible to group by Service Desk "Request type" field
- Fixed: Sub-task extender occasionally would not remove no longer matching sub-task from structure
- Fixed: Issues would not appear under Assignee when added from Search&Add panel
- Fixed: On macOS columns would overlap **+** (Add column) button
- Fixed: When clicking on the value of label-type custom field, transformation would be created around "labels" field instead

This version also includes performance improvements for static and generator-based structures.

7.27.3 Supported Versions

Structure 4.3 and all extensions support Jira versions from 7.2 to 7.5. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.2. Structure.Testy[498] extension, Colors[499] plugin, integrations with our partner add-ons should work with the new version.

496 https://wiki.almworks.com/display/structure2gmaster/Transformations
497 https://wiki.almworks.com/display/structure2gmaster/Filter
498 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
499 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.27.4 Installation and Upgrade

7.27.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for .Pages and Helper add-on, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.27.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x–4.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.27.5 Enterprise Deployment Notes

Structure 4.3 has a few changes in the core parts that could potentially affect performance and stability of the system.

7.27.5.1 Formulas and Attribute Subsystem

The Formula column is an important Structure feature that allows users to define and calculate arbitrary metrics for structures of issues. New aggregate functions and formula features in Structure 4.3 required significant changes to the formula engine and the attributes subsystem. If you use formulas and aggregate columns with big structures (10,000 issues or more), it makes sense to check their performance on a staging system before upgrading.

The expansion of formula feature may invite users to create new automated structures and running a formula with aggregation over those structures. On large Jira instances this may lead to increased server load, as formulas with aggregation may need to scan a lot of issues to calculate a total. It's a good idea to advise the users to calculate metrics on such large structures only if there's a real need. In most cases, structures can be made smaller.

7.27.5.2 Time Tracking Section Changes

Structure automatically sums up time tracking information and displays aggregate values in the time tracking section on the issue page. This feature has caused significant server load in cases when multiple users were actively working with issues from multiple big structures. In order to improve time tracking section performance in Structure 4.3 we have relaxed its security checks — now the time tracking section sums up the data for all issues in the current structure, including those that the current user cannot see because of issue security restrictions. This also means that all users will now see the same values in the time tracking section. If you rely on the Structure-adjusted time tracking section, please carefully assess these changes before upgrading.

7.27.5.3 Automation Engine Improvements

Structure 4.3 contains certain fixes and performance improvements in the automation engine. Large updates of large static structures, like moving hundreds of issues at once in a structure of tens of thousands (with no automation), should be processed much faster. Another change improves the performance of level-limited extenders, which should reduce server load during large automated structure regenerations. Because the automation engine is one of the most complex parts of Structure's core functionality, it makes sense to perform load and stress testing before upgrading.

7.27.5.4 Testing on Staging Environment

You can try load testing and stress testing Structure 4.3 and Jira on a staging environment before upgrading.

It makes sense to test the following parts:

- Open large structures and add multiple predefined columns from Calculated section. Do that from a number of client computers using different user accounts.
- Loading Jira's issue page for many issues by many users at once. (Use issues from projects that are enabled for Structure.)

The usual load and stress testing can also be applied.

Need help or have questions? Contact Tempo Support[500].

7.27.6 API Changes in Structure 4.3

7.27.6.1 Minor Java API Release

There are a few moderate changes coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.2+	16.6.0

500 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

See Structure API Versions(see page 1069) for full version information and downloads.

7.27.6.2 Compatible Changes in the Java API

Attribute API changes

Support for aggregation over subtree part has been added. Four subtree part kinds are supported:

1. `CHILDREN` – direct children only
2. `LEAVES` – leaf forest rows only
3. `STRICT` – full subtree without root node
4. `SUBTREE` – full subtree

Constants are declared in `CoreAttributeSpecs.Param`; behavior is described in `ReductionStrategy`. One can apply behavior to loader by subclassing `ReducingAggregateLoader` and overriding `ValueReducer.reduce` method (and other `ValueReducer` methods if convenient). Chosen subtree part is specified by `AttributeSpec` parameter 'type' (`CoreAttributeSpecs.Param.TYPE`) and defaults to `SUBTREE`.
List of subtree-part-aware loaders in api module:

- `LongSumLoader` (changed)
- `NumberSumLoader` (changed)
- `AbstractDistinctSumLoader` (changed)

To support this change some other changes have been made:

- Since distinct loaders can now accept parameters `DistinctAttributes.NON_UNIQUE_ITEMS_SPEC` has been deprecated and `DistinctAttributes` was marked as `@Internal`.
- Handy `SingleDependencyReducingAggregateLoader` (subclass of `ReducingAggregateLoader`) was added.
- Class `SecureSingleDependencyReducingAggregateLoader` aggregates values from subtree part, replacing value to `AttributeValue.undefined()` if user shouldn't have access to data by security reasons.
- `AttributeLoaderSecurity` encapsulates security checks. Can be used when `SecureSingleDependencyReducingAggregateLoader` isn't applicable.
- `AttributeValue.isEmpty()` is a shortcut for `!AttributeValue.isDefined() && AttributeValue.getLoaderData(Object.class) == null`
- `AttributeValue.ofNullable()` is a shortcut for `value == null ? AttributeValue.undefined() : AttributeValue.of(value)`
- `NumberAccumulator.toNumber()` is a shortcut for `NumberAccumulator.toValue().getValue()`
- `AbstractAggregateLoader.firstChildError()` method traverses errors through `AttributeLoader.Aggregate`. Used to transmit formula errors (other loaders currently don't supply error values).

`ForestSpec`-sensitive attributes

Attribute values can now depend on the forest spec for which the attribute is requested. Two new methods have been added to `AttributeContext` to support this:

- `getBaseForestSpec()` returns the `ForestSpec`, if there is one;
- `getBaseStructureId()` returns the structure ID part of the base forest spec.

Also there is a new version of `StructureAttributeSpec.getAttributeValues()`, marked `@Internal`.

The Notes column introduced in Structure 4.1 now uses the base forest spec instead of the `structureId` parameter, which has been removed.

Jira Service Desk Request Type support

A constant and a static method have been added to support the Request Type from Jira Service Desk as a separate item type:

- `CoreItemTypes.REQUEST_TYPE`
- `CoreIdentities.sdRequestType()`

Miscellaneous

- Added `GenerationContext.getForestSpec()`, which returns the forest spec being generated.
- `ValueFormat.ANY`, introduced in version 16.5.0, is now supported by the REST API.
- Added new versions of `ForestAccessCache.getInvisibleRows()` which accept `LongSizedIterable` as input. The old methods accepting `LongList` are now `@Deprecated`, kept only for binary compatibility.
- `ArrayForest.moveSubtreeAtIndex()` now returns -1 if no move is needed, i.e. the subtree is already at the given location.
- `@NotNull` and `@Nullable` annotations have been added in multiple places to clarify nullability contracts.

7.28 Structure 4.2 Release Notes

27th of August, 2017

Structure 4.2 adds major Formula language improvements, custom reporting period for Work Logged column, and a way to see large values that don't fit the column width. Other updates and improvements are also included.

Download the latest Structure and Extensions
(see page 1164) Structure Demo Server[501]

7.28.1 Version Highlights

- Formula Column language improvements.
- Filtering work logs by a custom period in the Work Logged column.
- Large texts or other values that don't fit the column width can now be displayed fully.

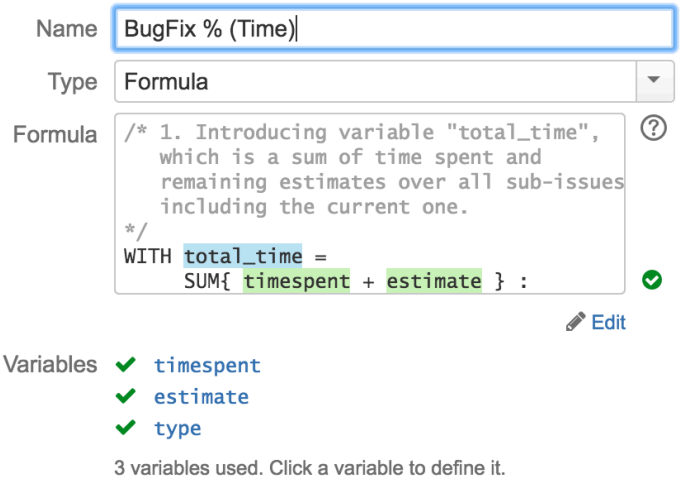
501 https://demo-structure.almworks.com/

7.28.2 Changes in Detail

7.28.2.1 Formula Language Improvements

There are a number of Expr language improvements delivered with this version.

First, we're adding **aggregate functions**, which are able to calculate a total value over sub-issues. In this version, we include only `SUM{}` aggregate function, but more aggregate functions will follow soon. But already with `SUM{}` only you can calculate a number of interesting metrics, such as WSJF or percentage of bugs in a structure. Here's a simple example of a formula that gives the number of bugs: `SUM{ IF(type=Bug;1) }`



Secondly, we're adding **local variables**, which is really helpful if the formula contains some expression multiple times.

Lastly, now you can write **comments** in a formula, which should make it is easier to read the formula later and make changes.

Remember that with Formula Column, it is possible to define a metric and share it with the team via Views(see page 505).

Documentation: Expr Language(see page 235), Aggregate Function Reference(see page 387)

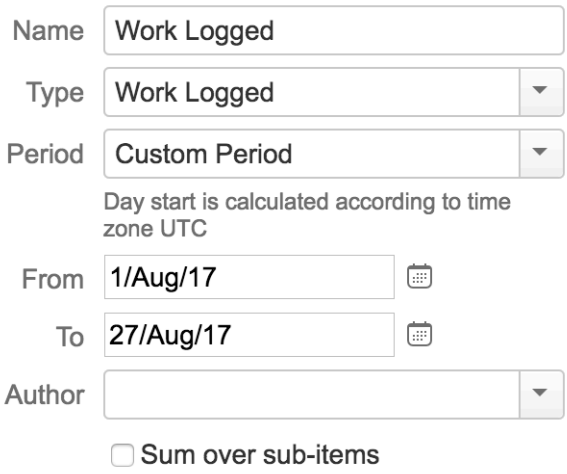
7.28.2.2 Custom Period in the Work Logged Column

Work Logged column displays total hours logged for an issue, with some additional filters. One of the filters selects only the work logs filed for a specific time period.

With Structure 4.2, it is possible to set an arbitrary period for Work Logged column (in addition to already available predefined periods like Today or This Year).

This addition option allows you to build task-based timesheet reports.

Documentation: Work Logged Column(see page 499)



7.28.2.3 Full Content Hover Box

When a field value is too large to fit into the cell, only a part of it is shown by Structure.

With version 4.2, it is possible to view the full content by hovering mouse pointer over the "more" sign (three vertical dots). This sign appears near the right edge of a cell if there's more to show.

This can come in handy if you do not want to click every issue to view their description in the issue details panel.

Documentation: Displaying Full Cell Content(see page 517)

7.28.2.4 Notable fixes and improvements

- The process of restoring Structure from backup now displays a progress bar and can be cancelled.
- Fixed: Changing filter text in the panel header does not result in change in filter results.
- Fixed: License installed on one node is not applied on another node in the cluster (JIRA Data Center).
- Fixed: Count leaves column has incorrect values when exported to Excel.
- Fixed: Values in Formula Cells with Duration format are missing when exported.
- Fixed: Potential data corruption during backup because of runtime exception.
- Fixed: Sub-task synchronizer doesn't add freshly created sub-tasks.
- Fixed: Time Tracking panel disappears when switching between issues on Internet Explorer.

The version also includes performance, reliability and other internal improvements.

7.28.3 Supported Versions

Structure 4.2 and all extensions support JIRA versions from 7.2 to 7.4.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

⚠️ Structure.Pages 1.2 or earlier are not fully compatible with Structure 4.2. If you're using Structure.Pages, make sure to also upgrade it to version 1.3 or later.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.1. Structure.Testy[502] extension, Colors[503] plugin, integrations with our partner add-ons should work with the new version.

7.28.4 Installation and Upgrade

7.28.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on JIRA and on Confluence side.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

502 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
503 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.28.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x-4.1 is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large JIRA, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.28.5 Enterprise Deployment Notes

Structure 4.2 has a number of changes that are particularly important for large installations and JIRA Data Center instances.

7.28.5.1 Events, Re-indexing and Experimental Feature

Over last few months, we've been working on tackling a race condition that may happen on a clustered JIRA. In short, Structure has its own means of inter-node communications, which is used to notify about issue changes. These notifications are important because changes made on one node may cause synchronizers or automation rules to be executed on another node. These rules or synchronizers often access JIRA Lucene index, to run a query, for example.

The problem is a race between our notification subsystem and JIRA's own notifications between nodes, which cause issue to be re-indexed on all nodes. If JIRA re-indexing is too late and our recalculation happens first, the users working with that node may see outdated or inconsistent data.

The solution to this problem required a lot of changes to Structure's event subsystem. Since it's a core component right in the middle of Structure's architecture, we approached its rollout carefully. Currently, the new event distribution approach for Data Center is an experimental "dark feature", which means that you need to turn it on explicitly, if needed.

To activate the updated "index monitoring" based event distribution subsystem on JIRA Data Center:

1. Verify that you need it. The cause for activating it could be the users complaining about not seeing the effects of actions of other users, or if your testing reveals this problem.
2. Set "`structure.delegatingItemTracker.enableReindexMonitor`" system property to "`true`".
3. Disable and then enable Structure. (Make sure there's enough time for disabling to propagate through all nodes.)
4. Verify: a new, undeletable system custom field called "**Structure Index Monitor**" should appear. This field will not have effect on issues, and it will disappear on its own if you turn this dark feature off.
5. Note that JIRA will tell you that a Full Re-index is required (since a new field added). Feel free to ignore this request, unless you also have other reasons for re-indexing.

In one of the upcoming versions of Structure, we'll enable this dark feature by default.

Takeaways:

- Current versions of Structure, including 4.2, in certain cases may show outdated data to some users on JIRA Data Center.
- The solution is currently an experimental feature, which can be enabled through a system property.
- There's an automatically created system custom field if this feature is enabled. It should not interfere with JIRA configuration.
- Even if the experimental feature is not enabled, there have been certain changes in a core subsystem in Structure.

7.28.5.2 Cancelling Structure Restore

Structure restore may take a long time, if there's a lot of Structure data (especially, history records). It may take an hour or more to restore Structure from backup, unless you decide to skip restoring history data.

With Structure 4.2, you can cancel the restore process if it takes too long. However, please be aware that after that you'll have some of the Structure data restored and some of it missing. You may need to communicate that fact to the users or adjust the access to Structure.

As an alternative, you can try Merge Structure operations, which works like Structure Restore but allows you to pick which structures and other data to restore.

7.28.5.3 Formulas and Attribute Subsystem

Attribute subsystem in Structure is responsible for delivering column values to the grid in Structure Widget, and for a few other things. In Structure 4.2 there were a few internal improvements in the Attributes subsystem, which may affect performance and functionality. (Actually, some of the changes should increase performance.)

One of the big features that we're working on over last several releases is Formula Column. This is an important addition to Structure that brings capability to define and run arbitrary metrics over structures of issues. Structure 4.2 introduces important additional functionality to formulas.

The expansion of formula feature may invite users to create new automated structures and running a formula with aggregation over those structures. On large JIRA instances this may lead to increased server load, as formulas with aggregation may need to scan a lot of issues to calculate a total.

Previously, we've added a hard limit on a structure's size (100,000 rows – can be adjusted). Calculating a total over 100,000 issues would require JIRA to retrieve that many issues from the database, which is not going to be quick and would add a load on JIRA and the database. It's a good idea to advise the users to calculate metrics on such large structures only if there's a real need. In most cases, structures can be made smaller.

7.28.5.4 Testing on Staging Environment

You can try load testing and stress testing Structure 4.2 and JIRA on a staging environment before upgrading.

It makes sense to test the following parts:

- Open large structures and add multiple predefined columns from Calculated section. (See the description above about Formula Column feature.) Do that from a number of client computers using different user accounts.
- Open the same structure in different browsers, having logged in under different users. Make sure the users go to different nodes in cluster. (Use direct node's addresses if necessary.) Make updates to a test structure

and test issues in one browser and watch if changes are picked up (without refreshing the page) on another browser. (You might need to switch to the other browser to see the changes.)

The usual load and stress testing can also be applied.

Need help or have questions? Contact Tempo Support[504].

7.28.6 API Changes in Structure 4.2

7.28.6.1 Minor Java API Release

There are a few moderate changes coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

JIRA Version	New API Version
7.1+	16.5.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.28.6.2 Compatible Changes in the Java API

Bulk attribute loading

`BulkAttributeLoader` is a new optional interface for attribute loaders. If your loader implements it, Structure will call its `preload()` method once for each attribute calculation, allowing you to perform efficient bulk calculations, e.g. run a single JQL query for all issues in the forest.

Three new methods have been added to `CompositeAttributeLoader` to support bulk attribute loaders: `hasBulkLoaders()`, `isBulkLoader()` and `preload()`.

Progress and cancellation for long-running processes

`ProgressGauge` is a new interface allowing you to track the progress of a long-running operation and gracefully cancel it from a different thread. Currently the only long-running operation with progress and cancellation support is `RestoreOperation`, which has a new method, `getProgressGauge()`. We plan to add progress and cancellation to `BackupOperation` in the future.

`ProcessHandleManager` implementation has been updated to support progress reporting and cancellation, both within a single JIRA instance and between different nodes in a Data Center environment. The following API methods have been added for this purpose:

- `ProcessInfo.getActivity()`
- `ProcessInfo.getPercentComplete()`
- `ProcessInfo.cancel()`
- `ProcessFeedback.isCancelled()`

504 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

- `ProcessUIController.setProgress()`

`SQuerySkeletonFactory` interface (experimental)

`SQuerySkeletonFactory` is an experimental extension point which lets developers add new query types to Structure. It is a work in progress, and it does not have client-side support yet.

Miscellaneous

- Added two new constants referring to the icon attribute, `CoreAttributeSpecs.ICON` and `CoreAttributeSpecs.Id.ICON`.
- Added a new attribute value format, `ValueFormat.ANY`.
- Added `getProjectsForCurrentUser()` method to `StructurePluginHelper`.
- Added `CoreSemantics.GROUP`, a new semantics for group rows created by groupers.
- Added `I18nText.setArguments(Object[])`, deprecated `setArguments(String[])`.
- Added `StructureUtil.isSubMap()` method.

`@Internal` components and classes

`ForestAccessCache` is the component used by Structure and Structure-based add-ons to check users' access to JIRA issues and Structure rows. It has been moved to the API and marked `@Internal`.

`SimpleAttributeProvider` is an abstract base class for several `AttributeLoaderProvider` implementations in Structure. Its implementation has been changed to better support optional spec parameters, and the class has been marked `@Internal`.

7.28.6.3 Item Type API Example

We have added a new sample plugin, `custom-itemtype`, which contains the following:

- a StructureItemType implementation for JIRA projects, complete with change tracking and access checks;
- an AttributeLoaderProvider implementation, which loads project names and icons;
- a StructureGenerator.Inserter implementation, which adds projects from one or more project categories.

You can download the plugin and its source code from API Usage Samples(see page 1152).

7.29 Structure 4.1 Release Notes

19th of June, 2017

Structure 4.1 introduces Notes column for additional comments on any items in the structure and a Full Screen mode. It also contains other improvements, important updates and a security fix.

Download the latest Structure and Extensions
(see page 1164) Structure Demo Server[505]

505 https://demo-structure.almworks.com/

7.29.1 Version Highlights

- Notes column lets you store additional text for any issue, folder or other item in a structure without creating custom fields or otherwise altering JIRA configuration.
- Grouping by version name and sprint name is useful for analysis when you have multiple projects with synchronized versions or sprints.
- Full-screen mode gives you more screen space for data and removes visual noise.

Structure 4.1 also contains a critical security patch.

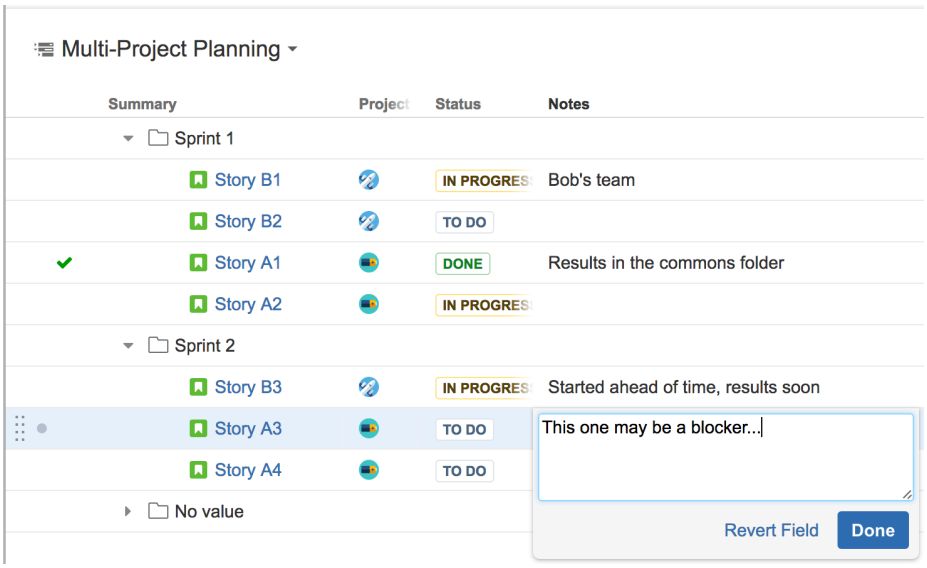
Upgrade is required for all JIRA instances running JIRA 7.1.0 and later. If you are using a previous version of JIRA, please make sure to upgrade to a version with the latest security patch – Structure 3.5.1(see page 880) for JIRA 7.0.x, Structure 3.3.5(see page 897) for JIRA 6.4.x, and Structure 3.2.3(see page 904) for JIRA 6.3.x.

> You need to have a Structure license with active maintenance (expiring not earlier than June 20th, 2017) to run use the new version or the patches.

7.29.2 Changes in Detail

7.29.2.1 Notes Column

A new type of column - Notes - can now be added to a structure. It allows leaving additional notes in a structure for any issue or folder, without the need to create a custom field. Notes can be used to add status updates without disturbing the usual workflow for other users.



Each structure has its own notes, even for the same items.

Notes are stored in the Structure's database, so even if you remove Notes column, they are not gone – you can add the column back later and the notes you left are preserved.

Documentation: Notes Column(see page 470)

Keyboard shortcut: **Ctrl-Shift-|** (Windows) or **Command-|** (Mac) to add a column

7.29.2.2 Grouping by Version and Sprint names

It is now possible to group by version name (from Fix Version field) and sprint name.

This comes useful when you have multiple projects with a synchronized schedule, having the same versions and same sprints. If you combine issues from those projects together and group them by Sprint or by Fix Version, you'll get multiple groups for each sprint – because, from JIRA's standpoint, those are different things.

Grouping by the name of version or sprint is a case of grouping by text value (Text Attribute grouper is used). That grouper just considers the text value – in our case, the name of the sprint or version.

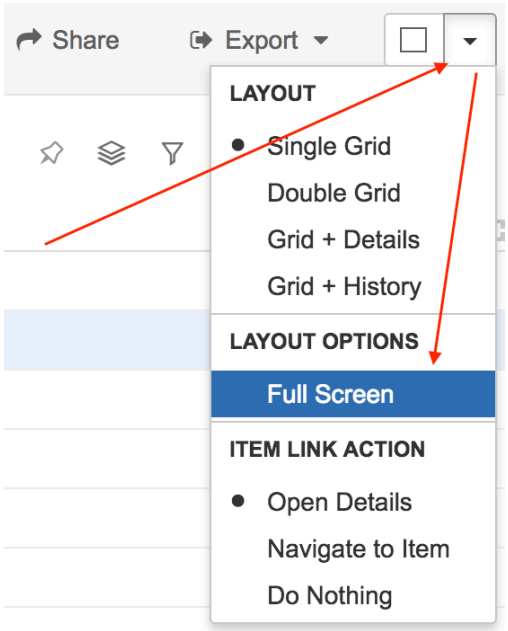
7.29.2.3 Full-Screen Mode

You can now expand Structure board to take the full screen, hiding JIRA header and, optionally, collapsing Structure toolbar. If you combine this with your browser's Full-Screen mode, you'll get maximum screen real estate dedicated to your tasks in a structure.

This is useful for presentations, using Structure board as a dashboard, working with large structures or just removing the visual clutter to concentrate on work.

Documentation: Full Screen Mode(see page 522)

Keyboard shortcut: **Z**



7.29.2.4 Structure Size Limit

We have introduced a hard limit on the structure size to safeguard against misconfigured synchronizers and imports. The limit is 100,000 rows maximum in a single structure, and it can be adjusted by a JIRA administrator.

A user who has Permission to Manage Synchronizers(see page 991) can inadvertently configure a synchronizer to pull in all issues into a single structure. That's not a problem unless JIRA instance is large. A structure with more than 100,000 issues in it may slow down JIRA, so we introduced this hard limit to protect the whole instance from erroneous actions of one user. Any action that would bring structure size over the limit would be denied.

Note that this limit only affects "static" structure, which does not include generators' output. Therefore, it protects from runaway synchronizers and manual changes. Generators have their own configurable limits and protections.

The hard limit may be adjusted by setting a system property and re-enabling Structure.

Documentation: Advanced Parameters(see page 856)

7.29.2.5 Security Patch

We have addressed a critical security issue, affecting all Structure versions starting with 3.0.

Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Security patches are provided separately for previous Structure versions, compatible with JIRA versions 6.3 and later.

7.29.2.6 Notable fixes and improvements

- Additional caching of sub-task links and issue links, improving performance of generators.
- "S" accelerator key removed from JIRA menu to avoid conflicts with Submit keyboard shortcuts in JIRA.
- Reduced number of log messages about deleted synchronizer when history is accessed.
- Fixed: Problems when deleting large structures from the user interface.
- Fixed: "After Filtering" option in aggregating columns is ignored by Export (both to Excel and Printable page).
- Fixed: Structure Gadget adds the displayed structure to Recent Structures list.
- Fixed: "Number of Leaves" column has no values exported in Excel.

7.29.3 Supported Versions

Structure 4.1 and all extensions support JIRA versions from 7.1 to 7.3.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

This release is backwards-compatible with Structure 3.4–4.0. Structure extensions – Structure.Pages[506], Structure.Testy[507], as well as Colors[508] plugin and other plugins that integrate with Structure 3.4 or later should work with the new version.

7.29.4 Installation and Upgrade

7.29.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.29.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x-4.0 is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large JIRA, consider turning off "**Backup History**" option to avoid long backup process.

---

506 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
507 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
508 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.29.5 Enterprise Deployment Notes

Compared to Structure 4.0, Structure 4.1 has a couple important additions on the server side.

7.29.5.1 About Structure Size Limit

As described above, by default the maximum number of rows that can be placed into a structure manually, via synchronizer, via import or via API is 100,000. We introduced this limit following a couple of cases where JIRA performance degraded due to misconfigured synchronizers. This is an additional safeguard for large JIRA instances.

However, there might be legitimate cases where a structure needs more than 100,000 issues in it and JIRA environment is powerful enough to work with it. In that case, the limit can be lifted as described in Advanced Parameters(see page 856).

7.29.5.2 New Table for Per-Structure, Per-Item Properties

With the introduction of Notes column, described above, we have added AO_8BAD1B_S_ITEM_PROPERTY table to the database schema. This table stores primary data (not stored anywhere else in JIRA).

The data from this table is exported along other structure data when you do Structure Backup(see page 962) in a human and machine readable form. The table is also exported along with all Structure data during JIRA backup.

We don't expect considerable database load on this table.

7.29.5.3 Testing on Staging Environment

There are no particular special areas of interest for load testing and stress testing Structure 4.1. We advise running the same testing procedures as you've done for previous upgrades.

Given the changes described above, you can also test the following:

- Try creating a synchronizer that inserts all issues in JIRA into a test structure. Use Filter synchronizer and JQL "project is not empty". It should succeed up to 100,000 issues.
- Try using a moderately large structure (up to 10,000) rows, add Notes column, enter some values and sort by it. That would place some load on the database to download values for all items.

Need help or have questions? Contact Tempo Support[509].

7.29.6 API Changes in Structure 4.1

7.29.6.1 Minor Java API Release

There are a few moderate changes coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

509 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

JIRA Version	New API Version
7.1+	16.4.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.29.6.2 Compatible Changes in the Java API

New methods in export API

Two new methods have been added to the export API to support the "after filtering" option in aggregated columns – `ExportRequestContext.requireAttribute(AttributeSpec, boolean)` and `ExportRow.get(AttributeSpec, boolean)`.

Another new method, `ExportRequestContext.getForestSpec()`, gives the export renderer provider access to the actual `ForestSpec` being exported.

`ItemTypeRegistry` interface

`ItemTypeRegistry` allows you to convert string item type IDs like `"com.almworks.jira.structure:type-issue"` to numeric IDs for more efficient storage and transport. Structure uses this registry internally, and now it's been made public.

Please note that the numeric IDs generated by `ItemTypeRegistry` are valid only on the JIRA instance they were generated on (including JIRA Data Center – they are the same for all nodes in a cluster). You cannot transfer them between different JIRA instances, you'll need to use the long string IDs for that.

Description property in `GeneratorPreset`

We've added optional descriptions to generator presets, so there's a new constructor `GeneratorPreset(label, description, parameters)` and a new method `GeneratorPreset.getDescription()`. Descriptions, if present, are shown as tooltips in the user interface.

Miscellaneous

- Added `equals()` and `hashCode()` implementations to `HistoryEntry` and `HistoryEntry.Change`.
- Added `info()` methods to `ConsiderateLogger`.
- Added `encodeURIComponent()` to `StructureUtil`.

7.29.7 Structure 4.1.1 Release Notes

20th of October 2017

Structure 4.1.1 is a security patch release.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[510]

510 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.29.7.1 Patch Release

This is a patch release based on Structure 4.1. We have addressed a minor security issue, affecting all Structure versions starting with 4.1 for those who use Notes column. Details of the issue are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Upgrade is required for all JIRA instances running Structure version 4.1 and using JIRA 7.1.x.

If you're using JIRA 7.2 or later, please upgrade to Structure 4.3[511] or later.

> ⚠ You should have Structure license with active maintenance (expiring not earlier than October 20th, 2017) to upgrade.

7.29.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.29.7.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
> - If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

511 https://wiki.almworks.com/display/structure2gmaster/Structure+4.3+Release+Notes

7.29.8 Structure 4.1.2 Release Notes

5th of May 2020

Structure 4.1.2 is a security patch release.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[512]

7.29.8.1 Patch Release

This is a patch release based on Structure 4.1. We have addressed a security issue that affects all Structure versions starting with 3.5. Problem was fixed in the latest (6.0) Structure.

Upgrade is required for all Jira instances running Structure version 4.1 and using Jira 7.1.x.

If you're using Jira 7.2 or later, please upgrade to Structure 5.1.1 or later.

You should have Structure license with active maintenance (expiring not earlier than May 6th, 2020) to upgrade.

7.29.8.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.29.8.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x and 4.x is simple:

512 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.30 Structure 4.0 Release Notes

26th of April, 2017

Structure 4.0 brings Excel-like functionality to JIRA with the new Formula column.

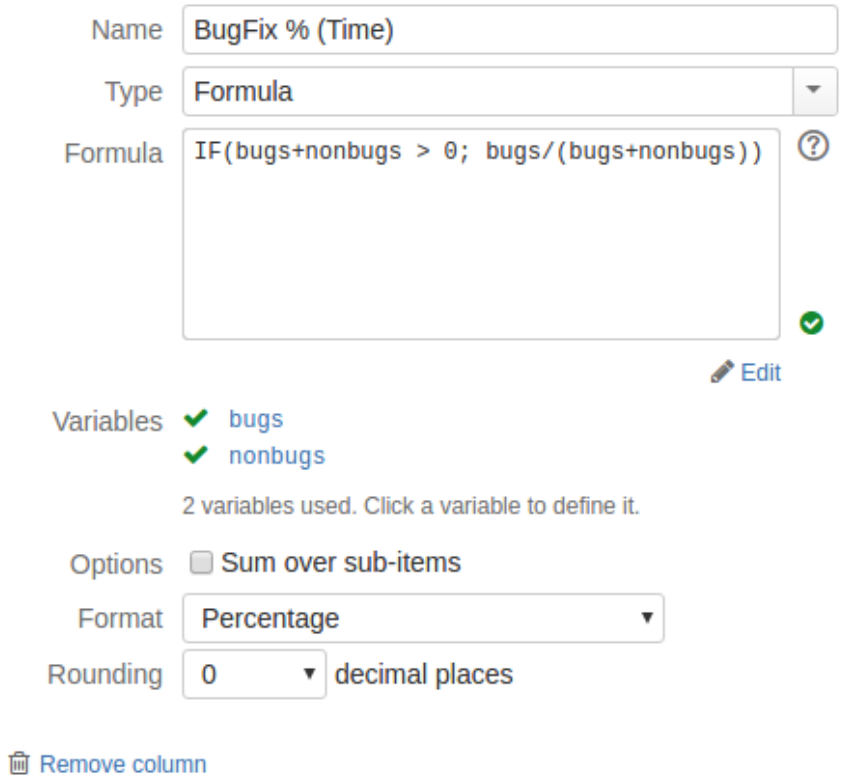
Download the latest Structure and Extensions
(see page 1164)Structure Demo Server[513]

7.30.1 Version Highlights

- Formula column
- Work Logged column for a specific user
- Making content static when copying or cloning a structure
- Exporting structures to other add-ons

513 https://demo-structure.almworks.com/

7.30.2 Changes in Detail



7.30.2.1 Formula Column

With Formula column, any user can perform some calculation on every item in a structure and display the result in the Structure grid. It is very similar to how formulas are used in Excel or other spreadsheet software.

To add a formula to your Structure view, click "**+**" at the top right corner of the grid, just like when adding a new column of other type. Scroll down to the bottom of the list and select "**Formula...**" to write your own calculated expression, or select one of the metrics from "**Calculated**" section.

When configuring Formula column, you can define the calculated expression using Expr Language(see page 235). It allows you to use values coming from issue fields, Structure-provided attributes (like totals) and other formulas. The values can be combined together, using arithmetic operators and more than 70 functions, to arrive at a single numeric or text value. Not only is that value displayed in the Structure grid, but it can also be used to sort the structure (just click on the column header) or to group items with the same value together, using Text Attribute grouper(see page 864).

The potential applications for Formula column are truly limitless – from all kinds of metrics to status reports to cramming different information into a terse text. Structure 4 comes with just a few examples like "Percentage of Bugs" and "Weighted Shortest Job First" metric, but any user (not necessarily a JIRA administrator!) can set up their own formulas.

Documentation: Formula Column(see page 463), Expr Language(see page 235), Expr Function Reference(see page 345)

Keyboard shortcuts:

- **Ctrl+|** (or **Command+|** on Mac) to add a new column.

- **Shift+Enter** when editing a formula to have it saved and verified; hit **Enter** to continue editing.

7.30.2.2 Work Logged by User

The Work Logged column has been around for a while already – it shows total amount of time logged during a specific period (for example, during last week).

In Structure 4, this column has an additional "**Author**" option. By default, it is empty and the column shows the total work during the selected period for all users – as it did before. If a user is selected, then only work performed by that user is considered.

This lets you build a work distribution report for a small team – just add several Work Logged columns and select a specific user in each.

Oh, and, by the way, you can use the value from Work Logged column in a formula!

Documentation: Work Logged Column(see page 499)

7.30.2.3 Making Static Copy of a Structure

Manage Structure page in Structure 3 lets you create a copy of a structure. Additionally, you can opt to create clones of all issues in the structure and populate the new structure with clones. See Copying Structure and Cloning Issues(see page 548) for details.

Starting with Structure 4, you have an additional option that governs what happens with 2022-01-04_18-18-09_Generators(see page 864) and the dynamic content that they produce. (As a reminder, dynamic content is something that gets updated automatically when things change in JIRA. It is created by generators – see Generators(see page 140) feature description and Structure 3 Quick Start Guide(see page 864).)

You can decide to leave the same generators in the new structure (so they would provide similar content), remove generators altogether (which means removing dynamic content as well), or to remove generators but replace the dynamic content with a "static" copy. The latter option is especially interesting as it allows you to "freeze" a structure in time so that further changes in JIRA do not affect the composition of the structure.

Documentation: Copying a Structure(see page 546)

7.30.2.4 Exporting Structures to Other Add-ons

We have made it possible for other add-ons to provide additional options in the Export menu on the Structure Board.

At the moment, we have integrations with the following products:

- Pivot Report[514] add-on can be used for additional analysis of your issue data. The latest version already has integration with Structure, just open Structure Board and select **Export | Pivot Report**.
- Xporter[515] is a solution for exporting your issues to various formats, including Word documents and PDF. Starting with version 5.1, Xporter will add **Export | Xporter for JIRA** menu on the Structure Board that will let you create nice-looking documents with all issues from the structure. Headings in the document are arranged according to the hierarchy in the exported structure.

We are also working with a few other add-on vendors on more integrations.

514 https://marketplace.atlassian.com/plugins/biz.epicreport.epicreports/server/overview
515 https://marketplace.atlassian.com/plugins/com.xpandit.plugins.jiraxporter/server/versionhistory

7.30.2.5 Notable Fixes and Improvements

There are a number of issues addressed and improvements added in Structure 4.0. The following are worth mentioning:

- Fixed: "Show all sub-items of matching items" option for Filter Transformation doesn't work.
- Fixed: JQL Completion doesn't work in the query input.
- Fixed: "Exclude duplicates" setting is ignored during Export.
- When migrating from Structure 2.x, avoid long-running backups by turning off the "Backup History" option.

7.30.3 Supported Versions

Structure 4.0 and all extensions support JIRA versions from 7.1 to 7.3.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

This release is backwards-compatible with Structure 3.4–3.6. Structure extensions – Structure.Pages[516], Structure.Testy[517], as well as Colors[518] plugin and other plugins that integrate with Structure 3.4 or later should work with the new version.

7.30.4 Installation and Upgrade

7.30.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.30.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large JIRA, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

---

516 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
517 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
518 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.30.5 Enterprise Deployment Notes

Structure 4.0 does not have a lot of core changes compared to Structure 3.6.

7.30.5.1 Formula Column Notes

The largest feature – Formula Column – is completely new and will not affect how the rest of the product is working. However, the feature itself is open to all users – anybody can add a formula and start calculating the values for structures. The Expr language engine is very fast, however, a lot of calculations on large structures may result in general slowdown due to constant retrieval of issue values. This is an unlikely event though, and if there's a really large structure that could cause such trouble (100,000 issues or more), it would produce the load on JIRA server regardless of the Formula column.

7.30.5.2 Access to User Properties

Structure 4 has the ability to extract custom user properties, defined in the JIRA's user management section. In a vanilla JIRA installation, there are no properties, but one can use them for defining, say, the hourly rate of a user or a category of a customer. These properties can be used in Formula columns, based on some user field.

The side effect of this improvement is that user properties become visible to everybody, because anyone can use Formula or Attribute column to display them. If you'd like to keep the users from gaining access to these user properties and disable attributes that provide them for the formulas, you can turn on `structure.userPropertyAttribute.disable` system property. (Add `-Dstructure.userPropertyAttribute.disable=true` to startup script and use `System.setProperty("structure.userPropertyAttribute.disable", "true")` in Script Runner on a running JIRA, then re-enable Structure.)

7.30.5.3 Copying and Cloning Structures

The updated Copy / Clone feature provides more opportunities to start cloning a large number of issues. Like in other cases, this may be a problem if a structure is large enough or has an unbounded generator that produces a lot of issues.

The feature has sufficient warnings to the user, notifying them about the number of issues that are about to be cloned. If, however, you need to turn Copying/Cloning off for your JIRA instance, please let us know and we'll provide instructions about how to do that.

7.30.5.4 Backup

Please create Structure backup before upgrading. Turn off Backup History option unless you really need it.

7.30.5.5 Testing on Staging Environment

You can try load testing and stress testing Structure 4.0 and JIRA on a staging environment before upgrading.

It makes sense to test the following parts:

- Open large structures and add multiple predefined columns from Calculated section. (See the description above about Formula Column feature.) Do that from a number of client computers using different user accounts.
- Run a copy & clone of a large structure with dynamic content.

The usual load and stress testing can also be applied.

Need help or have questions? Contact Tempo Support[519].

7.30.6 API Changes in Structure 4.0

7.30.6.1 Minor Java API Release

There are a few moderate changes coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

JIRA Version	New API Version
7.1+	16.3.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.30.6.2 Compatible Changes in the Java API

`BigDecimal` and `BigInteger` support for NUMBER attributes

Number attributes (see `ValueFormat.NUMBER`) are allowed to provide instances of `BigInteger` and `BigDecimal` as a result. The appropriate support was added to `NumberAccumulator` for calculating totals. We're limiting the precision of the numbers to `MathContext.DECIMAL64`.

Note that if your code used NUMBER attributes before and assumed that the result is either `Long` or `Double`, this assumption may be incorrect now.

Additional methods for formatting data in export API

The following methods were added to provide more power to the exporting code:

- `ExportColumn.setRounding()`
- `PrintableColumn.setDurationAsCalendarTime()`

Miscellaneous

- Additional factory methods in `CoreIdentities`.

7.30.6.3 Web Items

Extendable Export Menu

We have added a new web item location that lets you insert web items into Structure's Export menu.

519 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

Location name: `structure.export.menu`

To add a menu item, declare a `<web-item>` in that location and declare the following sub-elements:

- `label` - the name of the menu item, supports i18n, mandatory.
- `link` - the URL of the target page, mandatory. You can use `${forestSpec}` in the URL, which will be replaced with the URL-encoded JSON representation of the forest spec currently opened. If your URL does not have `${forestSpec}` in it, then forestSpec will be added to the URL as a query parameter.
- `tooltip` - the tooltip shown when mouse hovers over the item, supports i18n, optional.
- `styleClass` - a space-delimited CSS class names for the <i> icon element before the name, optional, defaults to a Structure-provided icon.
- `weight` attribute - defines the order of appearance of the links, optional.
- `condition` / `conditions` - can be used to display the links conditionally.
- `context-provider` - can be used to add additional variables to the link labels or URL.

7.31 Structure 3.6 Release Notes

4th of April, 2017

Structure 3.6 is another step towards integration with third-party add-ons. Now Structure supports grouping by Account field from Tempo Timesheets[520]. Apart from that, the release contains a number of fixes and smaller improvements.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 909) before upgrading.

Download the latest Structure and Extensions
(see page 1164) Structure 3 Demo Server[521]
Structure 3 Quick Start Guide
(see page 870)

7.31.1 Version Highlights

- Grouping by Account field from Tempo Timesheets
- Support for Portfolio 2.2.3 and 2.2.4

7.31.2 Changes in Detail

7.31.2.1 Grouping by Account

When building your structures you can now easily track progress and see distribution of time spent, remaining estimates and other values by Accounts that you have set up in Tempo Timesheets. To do that, simply add a Group Generator for Tempo Account, select if you want to see closed and archived accounts and if you want to be able to move issues between accounts in structure. You can use this grouping both in Automation (when building a structure) and in Transformations.

520 https://marketplace.atlassian.com/plugins/is.origo.jira.tempo-plugin/server/overview
521 https://demo-structure.almworks.com/

Structure supports Tempo 8.2.4/Tempo Account 3.0.7.

7.31.2.2 Support for Portfolio 2.2.3 and 2.2.4

The previous version introduced support for Portfolio Parent links[522] (both Group and Extend). This version adds compatibility with the latest versions of Portfolio.

7.31.2.3 Notable Fixes and Improvements

The following issues have been addressed in Structure 3.6:

- Fixed: Structure Gadget in Confluence becomes blank and cannot be configured.
- Fixed: Groups are grayed out, when applying Filter transformations.
- Fixed: Changes made to generators parameters are sometimes not applied.
- When cloning a structure with Automation, statistics is shown for the number of static items and items added dynamically.

The new version also contains other bug fixes and improvements.

7.31.3 Supported Versions

Structure 3.6 and all extensions support JIRA versions from 7.1 to 7.3.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

This release is backwards-compatible with Structure 3.5, so all Structure extensions – Structure.Pages[523], Structure.Testy[524], as well as Colors[525] plugin and other plugins that integrate with Structure 3.5 – should work with the new version.

7.31.4 Installation and Upgrade

7.31.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

522 http://almworks.com/blog/2017-01-track-portfolio-plans-real-time.html
523 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
524 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
525 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.31.4.2 Upgrading Structure

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large JIRA, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Upgrade Structure.Testy[526] and Structure.Pages[527] add-ons if you're using them.
5. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.31.5 Enterprise Deployment Notes

Structure 3.6 has a few changes in the core parts that could potentially affect performance and stability of the system. The changes are part of the ongoing improvement to optimize memory and database footprint by having a procedure of removing unused "rows" from Structure. This optimization itself is not yet turned on in Structure 3.6.

Rigorous testing has been carried out on the release candidate. The suggestions below provide further means to assure seamless deployment.

7.31.5.1 Backup

Please create Structure backup before upgrading. Turn off Backup History option unless you really need it.

7.31.5.2 Testing on Staging Environment

You can try load testing and stress testing Structure 3.6 and JIRA on a staging environment before upgrading.

It makes sense to test the following parts:

- Loading JIRA's issue page for many issues by many users at once. (Use issues from projects that are enabled for Structure.)
- Loading different structures and queries on the Structure Board by many users at once.
- Creating structures, populating them with generators and then deleting them.
- Creating a structure, adding considerable amount of issues (for example, 10,000 issues) and removing them from the structure.

The usual load and stress testing can also be applied.

526 https://marketplace.atlassian.com/plugins/com.almworks.testy
527 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview

Need help or have questions? Contact Tempo Support[528].

7.31.6 API Changes in Structure 3.6

7.31.6.1 Minor Java API Release

With this release we add a few constants and methods to the Java API. The changes are backwards-compatible.

JIRA Version	New API Version
7.x	16.2.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.31.6.2 Compatible Changes in the Java API

Tempo Account support

We have added two constants and two static methods to support the new Tempo Account grouper and the corresponding built-in item type:

- `CoreStructureGenerators.GROUPER_TEMPO_ACCOUNT`
- `CoreItemTypes.TEMPO_ACCOUNT`
- `CoreIdentities.tempoAccount(int accountId)`
- `CoreIdentities.isTempoAccount(ItemIdentity itemId)`

New versions of `RowManager.findRows()`

We have added two new `findRows()` methods to the `RowManager` interface:

- `void findRows(ItemIdentity itemId, LongPredicate consumer)`
- `default void findRows(ItemIdentity itemId, LongConsumer consumer)`

These methods guarantee that all row IDs your `consumer` is given during the `findRows()` call can be resolved by `getRow(long)` without throwing a `MissingRowException`. You can use the first method to stop the scan early.

The original method `LongIterator findRows(ItemIdentity itemId)` has been converted to a default method. Please note that this method will scan all rows before returning and that the row IDs produced by the resulting `LongIterator` may get deleted after the `findRows()` call returns.

528 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.32 Structure 3.5 Release Notes

30th of January, 2017

Structure 3.5 is a major release that adds grouping by issue links, filtering by JQL or S-JQL in Structure Gadget and bidirectional links extender. This release also has quite a few other updates, important security fixes, performance and stability improvements.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 909) before upgrading.

Download the latest Structure and Extensions
(see page 1164) Structure 3 Demo Server[529]
Structure 3 Quick Start Guide
(see page 874)

7.32.1 Version Highlights

Structure 3.5 is a major update in the Structure 3 series. It contains several new features:

- Grouping by issue links
- Specifying JQL or S-JQL as a filter in Structure dashboard gadget
- Bidirectional issue link extender
- Default level limit for extenders

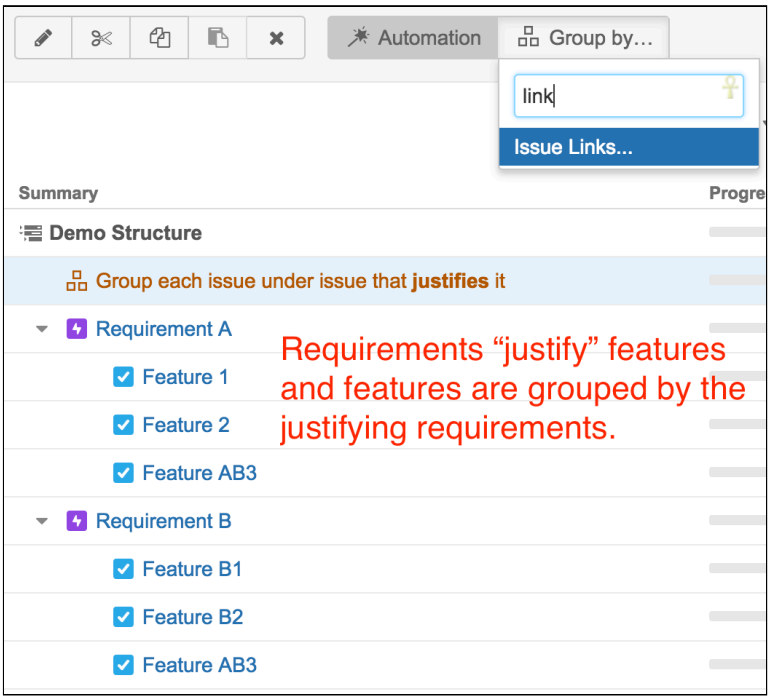
This version also contains a critical security patch.

Upgrade is required for all JIRA instances running Structure 3.0 and using JIRA 7.x. If you are using a previous version of JIRA, please make sure to install a version with a security patch – either Structure 3.3.4(see page 896) for JIRA 6.4.x or Structure 3.2.2(see page 903) for JIRA 6.3.x.

You need to have a Structure license with active maintenance (expiring not earlier than January 30th, 2017) to run use the new version or the patches.

529 https://demo-structure.almworks.com/

7.32.2 Changes in Detail



7.32.2.1 Grouping by Issue Links

There's a new grouper that can arrange a list of issues based on the incoming links of a certain type.

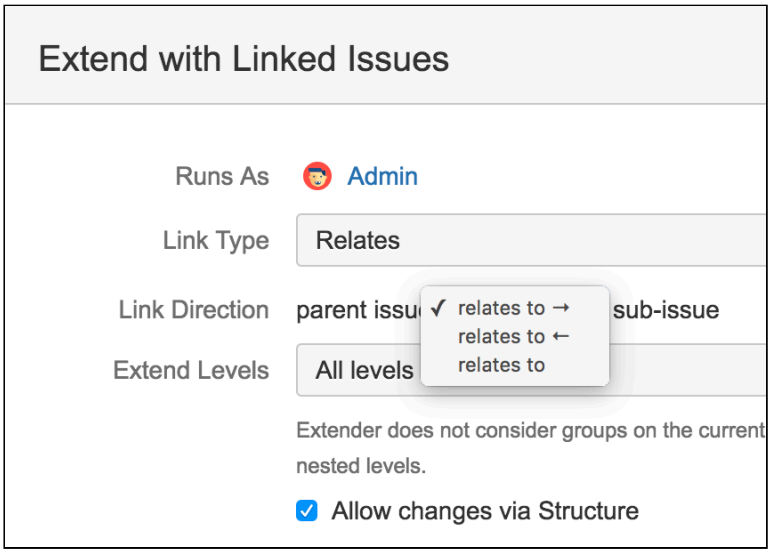
For example, if you have a list of Features and there are links between Requirements and Features, you can have the Features placed under the Requirement that is linked to it. If there are two incoming links from different Requirements, the Feature will be placed under both parent items.

7.32.2.2 Improved Filtering in Structure Gadget

Before Structure 3.5, the content shown in a Structure dashboard gadget could be filtered with a JIRA's saved filter. In this version we made it possible to also filter by a text query, a JQL query or an S-JQL query.

Support for S-JQL is especially important here, because it allows you to see a part of a big structure based on a hierarchical query. For example, you can display only "critical" issues and all their children, or a contents of a specific folder.

Documentation: Structure Gadgets(see page 565), S-JQL Cookbook(see page 429)



7.32.2.3 Bidirectional Issue Link Extender

Sometimes a link type does not have a "direction". An example is the standard JIRA's link type, "relates to". However, technically, there are two directions, but they are named identically. This was making it hard to build a hierarchy of related issues, because you would have to add a separate extender for each direction.

Starting with Structure 3.5, you can select both directions in the Links Extender, and it will produce a clean hierarchical view of the linked issues, regardless of the directions of the links.

The newly introduced Issue Links Grouper also support bidirectional operation.

7.32.2.4 Default Level Limit for Extenders

All extenders have "Levels" setting, which govern on what levels down the hierarchy the extender is applied.

Without a limit, the extender is applied at all levels under the parent that contains the extender. This sometimes led to problems where the extender was not configured properly and it resulted in structures 100 levels deep or more.

In Structure 3.5, the default setting for all extenders is to apply to 10 levels under. Should you need to increase that number, just edit the extender settings.

7.32.2.5 Security Patch

We have addressed two medium-to-critical security issues, affecting all Structure versions starting with 3.0.

Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

7.32.2.6 Notable Fixes and Improvements

The following issues have been addressed in Structure 3.5:

- JQL queries containing structure() function should execute faster.
- Backlog template now offers more fields to group by.

- Fixed: Structure does not react to issue updates after upgrading Universal Plugin Manager plugin.
- Fixed: Structure requiring unlimited commercial license in case one of the application licenses for JIRA is an evaluation license.

The new version also contains other bug fixes and improvements.

7.32.3 Supported Versions

Structure 3.5 and all extensions support JIRA versions from 7.0 to 7.3.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

> ⚠ Structure 3.5 is the last version of Structure to support JIRA 7.0. Structure 3.6 will most likely support JIRA versions starting from 7.1.

This release is backwards-compatible with Structure 3.4, so all Structure extensions – Structure.Pages[530], Structure.Testy[531], as well as Colors[532] plugin and other plugins that integrate with Structure 3.4 – should work with the new version.

7.32.4 Installation and Upgrade

7.32.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.32.4.2 Upgrading Structure

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
> - If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.

---

530 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
531 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
532 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

3. Upgrade Structure.Testy[533] and Structure.Pages[534] add-ons if you're using them.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.32.5 Enterprise Deployment Notes

Structure 3.5 is a fairly significant update. Some of the improvements are directly related to performance, stability and important Structure functionality.

7.32.5.1 Security Patches

Please note that the plugin contains security patches. It is important to upgrade to avoid potential data exposure. For the sake of safety, for time being we share the details only in the closed Structure Security Google Group[535].

If you cannot upgrade to Structure 3.5 because you have JIRA 6, please check out Structure 3.3.4(see page 896) and Structure 3.2.2(see page 903).

7.32.5.2 Plugin Start-Up Sequence

We have made further improvements to the plugin start-up sequence, where it needs to orchestrate the components start-up with JIRA start-up. Over time, this has proven to be tricky, but our current solution should be very robust. The improvements in Structure 3.5 concern the case where the UPM (Universal Plugin Manager) plugin is updated. This causes the whole plugin system to restart, and it is yet another case with different JIRA behavior.

Although we are pretty sure in the current solution, keep an eye out at the start of JIRA or when UPM is upgraded, watch the logs, and if there are any warnings or errors, please let us know.

7.32.5.3 Automation Engine Locking

We have made improvements to the Automation Engine, which should reduce the number of cases where you see timeouts in the logs ("cannot acquire lock in 30000ms") for various structures.

Please do not confuse this with other locks! JIRA has a lot of locks and many have 30 second timeouts. These locks in particular don't block other JIRA actions, but come into play when multiple users open the same structure (or when a single user opens the same structure in multiple browser windows). The locks protect JIRA server from doing the same work in parallel. If there is a timeout, it means that some structure is generated for a long time – for more than 30 seconds – so either Automation engine has to crunch a lot of issues, or the system is slow generally, or there is some configuration problem. In any case, you can inspect the structure, finding it by its ID, which is given in the same warning in the logs.

However, this change is in a critical and very complex part of the product, so there's a small risk of locking issues, related to the aforementioned warnings in the logs, structure lock-downs (when a structure is not displayed, the users just see empty content), or redundant calculations on the server resulting in higher server load.

7.32.5.4 Asynchronous Index Writing

Structure keeps an index of "which structures contain which issues". This index is not critical to operation and is used only sometimes – for example, when selecting which structure to display on the issue page. The index also ignores the generated content; it only works for issues added manually or by synchronizers.

---

533 https://marketplace.atlassian.com/plugins/com.almworks.testy
534 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
535 http://groups.google.com/group/structure-security-list

In some configurations, updating that index introduced unwanted latency, so we moved its updates to an asynchronous job, executed by a background thread. While this change is not likely to cause any trouble, the index updates may be delayed on a heavily loaded system and that may result in an incorrect structure being picked to show on an issue page.

7.32.5.5 Testing on Staging Environment

It is recommended to try Structure 3.5 on a staging environment before upgrading. The suggested tests are:

- Plugin lifecycle.
    - Installing and uninstalling. Disabling and enabling.
    - Upgrading UPM, upgrading JIRA Agile plugin.
    - Restarting JIRA. (There are no Data Center-specific changes in this version.)
- Try making changes to structures in rapid bursts. Watch out for JVM memory statistics and CPU consumption. See if there is a lot of activity even after the bursts stop.
- Check out generators that work with issue links – the Extender and the Grouper. Try using unidirectional and bidirectional configurations on large structures.
- Check Structure Gadget.
    - Check if it works on JIRA dashboard.
    - If you have Confluence instance connected and Structure gadget is used there, check that it works in your configuration.
- Check "Export" button, exporting to Excel and to Printable page. Try on large structure and under a user account without administrative privileges.
- Automation locking.
    - Try opening a structure with automation from multiple browsers. You can build a temporary structure with considerable work for generators – for example, inserting 5,000 issues and using Issue Links extender.
    - Try using transformations (such as filters) on a structure, along with opening that structure in other browsers.
- Automation load testing.
    - Try opening user structures that contain considerable amount of issues, 10,000 or more. Click grid header to make Structure sort structure by some field. Do that in multiple browser tabs and using different columns.
    - Try creating a new structure and populate it with the help of an Inserter (use Automation | + | Insert | JQL) and raise the issue limit to 10,000. Repeat this several times with different structures.
    - Watch log files for errors and warnings.
- Automation stress testing.
    - Emulate peak number of users opening the most popular structure.

Need help or have questions? Contact Tempo Support[536].

536 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

7.32.6 API Changes in Structure 3.5

7.32.6.1 Minor Java API Release

With this release we add a few constants and a static method to the Java API. The changes are backwards-compatible.

JIRA Version	New API Version
7.x	16.1.0

See Structure API Versions(see page 1069) for full version information and downloads.

7.32.6.2 Compatible Changes in the Java API

New constant in `StructureLicenseError`

- `JIRA_EVAL_LICENSE_MISMATCH`

New constants in `CoreStructureGenerators`

- `GROUPER_AGILE_SPRINT`
- `GROUPER_LINKS`

New method in `StructureUtil`

- `getBooleanSystemProperty(String key, boolean defaultValue)`

7.32.7 Structure 3.5.1 Release Notes

19th of June 2017

Structure 3.5.1 is a security patch release for JIRA 7.0.x.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[537]

7.32.7.1 Patch Release

This is a patch release based on Structure 3.5.0. We have addressed a critical security issue, affecting all Structure versions starting with 3.0. Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

537 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is required for all JIRA instances running Structure versions 3.0–3.5.0 and using JIRA 7.0.x.

If you're using JIRA 7.1 or later, please upgrade to Structure 4.1(see page 856) or later.

⚠ You should have Structure license with active maintenance (expiring not earlier than June 20th, 2017) to run the patches.

7.32.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.32.7.3 Upgrade

⚠
- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.32.8 Structure 3.5.2 Release Notes

ⓘ **5th of May 2020**

Structure 3.5.2 is a security patch release for Structure 3.5

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[538]

538 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.32.8.1 Patch Release

This is a patch release based on Structure 3.5.1. We have addressed a security issue that affects all Structure versions starting with 3.5. Problem was fixed in the latest (6.0) Structure.

Upgrade is required for all Jira instances running Structure versions 3.5.0 and using Jira 7.0.x.

If you're using Jira 7.1 or later, please upgrade to Structure 4.1(see page 856) or later.

> ⚠ You should have Structure license with active maintenance (expiring not earlier than May 6th, 2020) to run the patches.

7.32.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.32.8.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
> - If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.33 Structure 3.4 Release Notes

> ⓘ **12th of December, 2016**
>
> Structure 3.4 is a major release that adds quick transformations (a.k.a. quick filters), integration with Portfolio for JIRA and stable public API. It also has quite a few other updates, bug fixes, performance and stability improvements.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 909) before upgrading.

Download the latest Structure and Extensions
(see page 1164) Structure 3 Demo Server[539]
Structure 3 Quick Start Guide
(see page 882)

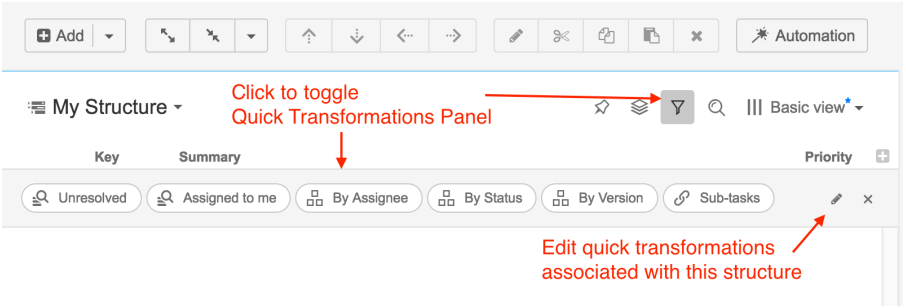
7.33.1 Version Highlights

Structure 3.4 is a major update in the Structure 3 series. It contains several new features:

- Quick Transformations
- Integration with Portfolio for JIRA
- Issue Details layout on the Project page
- Grouping by text fields
- Performance improvements
- Public API

Upgrade is recommended to all users.

7.33.2 Changes in Detail



7.33.2.1 Quick Transformations

Quick transformations, also known as "quick filters", are predefined transformations that may be configured by the structure owner and then quickly turned on, separately or in combination.

Previously, Structure had only two predefined quick transformations – Unresolved and Assigned to me. Now it is possible to define your own quick transformations and share them with the team.

Documentation: Quick Transformations(see page 220)

7.33.2.2 Integration with Portfolio for JIRA

We have added support for Parent Link field in Portfolio for JIRA.

539 https://demo-structure.almworks.com/

Now, when you have Portfolio installed, you get two new generators:

- An extender called **Child Issues (Portfolio)**, which adds "child" issues, as defined by Parent Link field, under their "parent" issues.
- A grouper called **Parent Issue (Portfolio)**, which goes the other direction – groups issues based on the value in their Parent Link field, which adds the parent issues to structure.

Both generators support updates, so you can change the Parent Link field by dragging and dropping issues.

This feature allows you to recreate in Structure the hierarchy you have in Portfolio and then build on top of it.

Supported Portfolio for JIRA versions: 2.1.5 or later.

7.33.2.3 Issue Details Layout on Project Page

Good news for those of you who prefer to use Project tab with Structure, rather than Structure Board – we have added "Grid + Details" layout to those available on the project page.

See Viewing Issue Details(see page 126) for documentation about this layout.

7.33.2.4 Grouping by Text Field

We have added a special generator that lets you group by a text attribute. A common example is grouping by the value of a custom text field.

To use this grouper, select **Automation | +**, then **Group**, and find **Text Attribute...** in the drop-down list. In the dialog that follows, select the desired attribute.

This grouper does not handle changes. You cannot move an issue from one group to another – you will need to update the field.

7.33.2.5 Performance Improvements

There are a few performance improvements in this version:

- We have replaced one of the core components, *RowManager*, that has been causing some trouble in the past. The new version should be much faster.
- There were significant fixes related to synchronizer performance.
- Backup creation with history is now much faster.
- We fixed some performance issues with structure() function used on Agile boards.
- Fixed performance issues caused by too many groups in the user directory.

7.33.2.6 Public API

We are finally releasing our stable API for Structure 3 series.

With the introduction of the new architecture earlier this year we had to make breaking changes to the API and took some time to work out the kinks and make it stable. Structure 3.4 ships with Structure API 16.0.0, which can be used in your custom plugins or from integration scripts.

We have documented the most important parts of the API but the documentation work still continues. You can expect that API coverage will increase with every new version.

Documentation: Structure Developer's Guide(see page 1029)

7.33.2.7 Notable Fixes and Improvements

The following other issues have been addressed in Structure 3.4:

- Structure restore writes progress percent to the logs, so JIRA administrator can check it.
- Fixed: Compatibility issues with Zephyr plugin.
- Fixed: Incorrect sorting direction when adding a sorter, in certain cases.

There are also other bug fixes and improvements.

7.33.3 Supported Versions

Structure 3.4 and all extensions support JIRA versions from 7.0 to 7.2. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

> ⚠ Structure 3.3 was the last version of Structure to support JIRA 6. Structure 3.4 supports JIRA versions starting from 7.0.

7.33.3.1 Structure.Pages Upgrade Required

Please note that if you're using Structure.Pages, you will need to upgrade it to version 1.2.0.

See Structure.Pages 1.2 Release Notes[540] for more information about updates in Structure.Pages.

7.33.3.2 Structure.Testy Upgrade Required

Please note that if you're using Structure.Testy, you will need to upgrade it to version 2.2.0.

Structure.Testy 2.2 includes new high-level API that makes it easier to integrate your scripts with Testy and update test statuses programmatically.

7.33.4 Installation and Upgrade

7.33.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

---

540 https://wiki.almworks.com/pages/viewpage.action?pageId=32213737

7.33.4.2 Upgrading Structure

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Upgrade Structure.Testy[541] and Structure.Pages[542] add-ons if you're using them.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.33.5 Enterprise Deployment Notes

Error rendering macro 'include'

com.atlassian.renderer.v2.macro.MacroException: No page title provided.

Structure 3.4 is a fairly serious update. A lot of code underwent refactoring, performance optimization and testing.

The following changes are quite important for the large-scale instances.

7.33.5.1 New implementation of RowManager component

We have rewritten the component that is central to the Structure's operation. The component's role is to store temporary rows as they are generated by Automation engine. The previous version was using "MapDB" library and offloading the stored values to disk. This turned out to be unreliable in some circumstances and under a heavy load.

The new version is much simpler and efficient, however, it does store this information in memory. The amount of data stored is not overly large, but current implementation never removes data from memory once it is stored. Therefore, on an active enough and large enough instance, this component may eventually grow to take a lot of memory resources or even cause an OutOfMemoryError.

Therefore, JIRA administrators on large instances are advised to keep an eye on the amount of Heap memory consumed by JIRA's Java process, and if it reaches what you'd consider to be a high threshold, you can clean up the memory taken by Structure with a simple operation: disable Structure plugin and then enable it back.

We are continuing our work on improving this component. Our next versions will contain improvements for this aspect.

541 https://marketplace.atlassian.com/plugins/com.almworks.testy
542 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview

7.33.5.2 Indexing on Data Center

We have identified a problem that can possibly manifest on JIRA Data Center cluster, when a node wants to recover indexes (from a downloaded index zip from another node), but it happens to be running a Structure synchronizer. As synchronizers used to take read lock on reindexing, they prevented full-stop reindex and index recovery from happening at the same time with synchronization. But on large instances a single synchronization job may execute for a long time – and this caused "Wait attempt timed out" error on the reindexing node.

The new locking mechanism, which is added in Structure 3.4, does not take read lock for more than a fragment of a second. While still ensuring that Structure's synchronizers and generators would execute based only on consistent query results, it allows JIRA full reindex or index recovery to happen at any time.

7.33.5.3 Improved Start-up Sequence

We have changed the way Structure plugin starts. Since we only support JIRA 7.0 or later with this version, we were to use some of the improvements Atlassian team prepared in JIRA 7 to increase reliability of the start sequence.

The related problem that happened to our customers in the past was that the plugin wouldn't start – it would wait for Active Objects component (provided by JIRA) to become available, but fail after 60 seconds of waiting. Usually, another attempt to manually start Structure would succeed.

With the new start-up sequence, the probability of such behavior should be less. Unfortunately, it's hard to tell precisely, because there's a dependency on JIRA subsystem. However, the plugin start-up has become much more predictable and repeatable.

7.33.5.4 Performance Improvements

Most of the performance improvements mentioned in the Release Notes above are quite important for large-scale instances.

7.33.5.5 Testing on Staging Environment

The following checks are suggested for Structure 3.4 on a staging environment with production data.

1. Start-up and shutdown.
    a. Try installing and uninstalling Structure several times. Watch the logs. On JDC, watch logs on every node.
    b. Try starting and stopping JIRA when Structure is installed. On JDC, try stopping/starting individual nodes and the whole cluster.
2. Index recovery on JIRA Data Center. Try forcing index recovery by manually delaying your node indexes. Before that, install synchronizers that are known to run for a long time – for example, a links synchronizer on all of your issues, assuming you have a lot of links in JIRA.
3. Automation load testing.
    a. Try opening user structures that contain considerable amount of issues, 10,000 or more. Click grid header to make Structure sort structure by some field. Do that in multiple browser tabs and using different columns.
    b. Try creating a new structure and populate it with the help of an Inserter (use Automation | + | Insert | JQL) and raise the issue limit to 10,000. Repeat this several times with different structures.
    c. Watch log files for errors and warnings.
4. Automation stress testing.
    a. Emulate peak number of users opening the most popular structure.

5. If you're using JIRA Software, try creating an Agile board that is based on `structure()` function (see S-JQL Cookbook(see page 429)). Make sure there is about 1,000 issues in the result. See how well Agile board loads.
6. Check **Administration | Structure | Maintenance** page. Try to run Structure backup and measure how much time does it take. Watch for errors in the logs.

Need help or have questions? Contact Tempo Support[543].

7.33.6 Structure 3.4.1 Release Notes

30th of December 2016

Structure 3.4.1 is a patch release based on Structure 3.4.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[544]

7.33.6.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has been changed in Structure 3!

This is a patch release based on Structure 3.4.

We have addressed a problem with Structure widget being shown on the issue page for issues from the projects for which Structure is not enabled.

Upgrade is recommended for all instances where Structure availability is set on the project basis.

7.33.6.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

543 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
544 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.33.6.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.33.6.4 Enterprise Deployment Notes

We now include an additional section in Release Notes that aims to address the concerns of deploying the upgrade at a large enterprise. In this section we will suggest how the changes in the new version may affect stability and performance of JIRA and provide ideas for testing the new version on a staging environment.

In terms of stability and performance, this patch does not bring significant changes compared to version 3.4.0.

7.34 Structure 3.3 Release Notes

9th of September, 2016

Structure 3.3 is a major release that adds support for structure migration between JIRA instances, extends S-JQL with ability to search for folders and other non-issue items, addresses compatibility issues with JIRA 7.2 and has other improvements and important fixes.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 909) before upgrading.

Download the latest Structure and Extensions
(see page 1164) Structure 3 Demo Server[545]
Structure 3 Quick Start Guide
(see page 889)

545 https://demo-structure.almworks.com/

7.34.1 Version Highlights

Structure 3.3 is a major update in the Structure 3 series. It contains several new features:

- Migration of Structure data between JIRA instances (also allowing partial restore from backup)
- S-JQL support for searching folders and other items
- Special filter that removes duplicates added by a combination of inserters and extenders
- Maintenance task that removes old structure change history (please check the defaults!)
- Swapping panels in two-panel mode
- JIRA 7.2 compatibility
- Significant fixes and improvements related to performance and troubleshooting

Upgrade is recommended to all users.

7.34.2 Changes in Detail

7.34.2.1 Structure Data Migration

It's now possible to import one or more structures from a different JIRA instance after you have imported projects with the JIRA's Project Import operation. Also, you can partially restore deleted or broken structures from a backup without affecting other structures.

Documentation: Structure Backup, Restore and Migration(see page 962)

> Migration used to be a Structure 2 feature, but it was disabled with the release of Structure 3, as we've moved the storage from a separate database to the JIRA main database. Now it is being reinstated.

7.34.2.2 Search for folders and other non-issue items with S-JQL

It is now possible to identify folders (whether specific folders by name or all folders) and other types of items in S-JQL expressions. This expands the power of S-JQL to all items – for example, it allows searching for a contents of a specific folder.

For example: `descendant of folder("next release")`

Documentation: Structured JQL(see page 428)

7.34.2.3 Special filter for removing duplicates from using Inserter and Extender

It is a typical setup when you use JQL Inserter to place all matching issues at some place in a structure, and then use an extender (for example, Links Extender) to build a hierarchy of relationships starting from the inserted issues. For example, to show all issues from a project with dependencies placed hierarchically, you'd add a JQL Inserter (`project = ABC`) and a Links Extender with (`depends on`) link type.

The problem with such setup is that an issue may get inserted once by the inserter, and then again added at a deeper hierarchy level by the extender.

To remove such duplicates, use a special "Remove Inserter/Extender Duplicates" filter.

Documentation: Inserter/Extender Duplicates Filter(see page 162)

7.34.2.4 Maintenance task to clear old change history

Daily maintenance now includes clearing up database space by removing old change history for structures. It is recommended to not disable this task to avoid database size buildup and reduction in activity streams querying performance.

Please check the default settings! By default, structure history that is both older than 30 days and less recent than 1,000 last changes (counted for each structure separately) will be removed daily. If you think that you need history of changes for a larger period, please review and make changes to the settings **immediately after the upgrade**!

Documentation: Automatic Structure Maintenance(see page 966)

7.34.2.5 Swapping panels in two-panel mode

When you have two structure panels open, you can now easily swap left and right panel by pressing **Alt+** keyboard shortcut.

7.34.2.6 JIRA 7.2 compatibility

Structure plugin and its extensions are now compatible with JIRA 7.2 and newer versions, including JIRA Data Center. The minimum supported version is JIRA 6.4.

7.34.2.7 Notable Fixes and Improvements

The following issues have been addressed in Structure 3.3:

- Added: Performance-related metrics, displayed to the JIRA admin, which would help ALM Works support diagnose performance-related issues faster.
- Added: Workaround for JIRA Data Center issues regarding cluster-based caches (JIRA issues JRA-62034[546] and JRA-62071[547]) that could cause JDC initialization failure.
- Fixed: Page freezes when removing extender in a structure with a lot of items.
- Fixed: Database troubleshooting page isn't accessible when database is locked.
- Fixed: Impossible to open issue page via clicking on the issue key on issue details panel.
- Fixed: Impossible to delete a structure that has slowly performing generators.
- Fixed: Slow activity stream queries.

7.34.3 Supported Versions

Structure 3.3 and all extensions support JIRA versions from 6.4 to 7.2. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

Structure 3.3 is the last version of Structure to support JIRA 6. Structure 3.4 will most likely support JIRA versions starting from 7.0.

546 https://jira.atlassian.com/browse/JRA-62034
547 https://jira.atlassian.com/browse/JRA-62071

7.34.3.1 Structure.Pages Upgrade Required

Please note that if you're using Structure.Pages, you will have to upgrade it to version 1.1.0.

7.34.3.2 Structure.Testy Upgrade Required

Please note that if you're using Structure.Testy, you will have to upgrade it to version 2.1.0.

7.34.4 Installation and Upgrade

7.34.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.34.4.2 Upgrading Structure

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Upgrade Structure.Testy[548] and Structure.Pages[549] add-ons if you're using them.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.34.5 Structure 3.3.1 Release Notes

22nd of September 2016

Structure 3.3.1 is a patch release based on Structure 3.3.

548 https://marketplace.atlassian.com/plugins/com.almworks.testy
549 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[550]

7.34.5.1 Patch Release

> ⚠ If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has changed in Structure 3!

This is a patch release based on Structure 3.3. We have addressed the following issues:

- Fixed: exception during Structure restore from version 2.x.
- Fixed: dropdowns on Agile boards stop working after clicking on an issue key.
- Fixed: Structure disappears on an issue page after clicking on an issue key.
- Fixed: dropdowns stop working after issue page refresh.
- Fixed: exception in DefaultWorkflowSchemeManager.
- Fixed: incorrect initialization leading to "dangerous call" StructureRuntimeException.
- Fixed: slow loading of Agile boards based on structure() function.
- Fixed: warning in the logs about Inserter/Extender Duplicates Generator.

Upgrade is recommended for all Structure users.

7.34.5.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.34.5.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
> - If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

---

550 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.34.6 Structure 3.3.2 Release Notes

3rd of October 2016

Structure 3.3.2 is a patch release based on Structure 3.3.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[551]

7.34.6.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has been changed in Structure 3!

This is a patch release based on Structure 3.3. We have addressed a problem with issue key link not working on the Issue Navigator page in Internet Explorer 11.

Upgrade is recommended for all Structure users who use Internet Explorer.

7.34.6.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.34.6.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

551 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.34.7 Structure 3.3.3 Release Notes

15th of November 2016

Structure 3.3.3 is a patch release based on Structure 3.3.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[552]

7.34.7.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has been changed in Structure 3!

This is a patch release based on Structure 3.3.

We have addressed a potentially critical problem on JIRA Data Center: a node may be unable to perform re-indexing due to a lock held by Structure. This may happen if a node is added to or removed from a cluster.

Upgrade is recommended for all JIRA Data Center instances. This version supports JIRA 6.4 — 7.2.x.

7.34.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.34.7.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

552 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.34.7.4 Enterprise Deployment Notes

We now include an additional section in Release Notes that aims to address the concerns of deploying the upgrade at a large enterprise. In this section we will suggest how the changes in the new version may affect stability and performance of JIRA and provide ideas for testing the new version on a staging environment.

In terms of stability and performance, this patch does not bring significant changes compared to versions 3.3.1 and 3.3.2. The only change was the fix for the reindex lockout problem, which was localized in the synchronization management code.

Additional testing and verification can be done on a staging environment if it is running Data Center as well. The possible testing scenario is:

- install a synchronizer on a test structure, verify that it is doing its job;
- add a node to the cluster, verify that reindex is possible on each node;
- locate the node that is running synchronization (use logs to see where the messages from synchronizers appear) and shut down that node;
- verify that the synchronizer continues running (it may take 5-10 minutes to switch to a new node) and that reindex is still possible on each node.

7.34.8 Structure 3.3.4 Release Notes

30th of January 2017

Structure 3.3.4 is a security patch release for JIRA 6.4.x.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[553]

7.34.8.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has been changed in Structure 3!

This is a patch release based on Structure 3.3.3. We have addressed two medium-to-critical security issues, affecting all Structure versions starting with 3.0. Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

553 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is required for all JIRA instances running Structure versions 3.0–3.3.3 and using JIRA 6.4.x.

If you're using JIRA 7.x, please upgrade to Structure 3.5(see page 874) or later.

⚠ You should have Structure license with active maintenance (expiring not earlier than January 30th, 2017) to run the patches.

7.34.8.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.34.8.3 Upgrade

⚠
- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.34.9 Structure 3.3.5 Release Notes

ⓘ **19th of June 2017**

Structure 3.3.5 is a security patch release for JIRA 6.4.x.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[554]

554 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.34.9.1 Patch Release

This is a patch release based on Structure 3.3.4. We have addressed a critical security issue, affecting all Structure versions starting with 3.0. Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Upgrade is required for all JIRA instances running Structure versions 3.0–3.3.4 and using JIRA 6.4.x.

If you're using JIRA 7.0.x., please upgrade to Structure 3.5.1(see page 880) or later version of Structure 3.5.x series.

If you're using JIRA 7.1 or later, please upgrade to Structure 4.1(see page 856) or later.

> ⚠ You should have Structure license with active maintenance (expiring not earlier than June 20th, 2017) to run the patches.

7.34.9.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.34.9.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
> - If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.35 Structure 3.2 Release Notes

18th of June, 2016

Structure 3.2 adds separate backup and restore for Structure data, improves user experience on the project page and contains important fixes and optimizations.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 909) before upgrading.

Download the latest Structure and Extensions
(see page 1164) Structure 3 Demo Server[555]
Structure 3 Quick Start Guide
(see page 899)

7.35.1 Version Highlights

Structure 3.2 is an incremental update in the Structure 3 series. It contains several new features, important improvements and bug fixes.

- Backup and restore Structure data
- Additional features on the project page
- Sequential index column
- Quick action lookup
- Structure gadget in Confluence
- Important fixes related to performance and stability

Upgrade is recommended; it is required if you experience any issues with the current version of Structure.

7.35.2 Changes in Detail

7.35.2.1 Backup and Restore

It's now possible to create a separate backup of the Structure data and later restore structures from backup. Although Structure data is also backed up when you make a full JIRA backup, the separate backup may be used to restore structures without affecting JIRA data or to export structures to other systems by reading the XML backup file.

We're still working on Migration feature, which will allow moving structures from one JIRA to another.

Documentation: Structure Backup, Restore and Migration(see page 962)

Backup and Restore used to be Structure 2 features, but they were disabled with the release of Structure 3, as we've moved the storage from a separate database into the JIRA main database. Now they are being reinstated.

555 https://demo-structure3.almworks.com/

7.35.2.2 Structure on the Project Page

We have listened to our users who were unhappy with how Structure worked on the project page and we have made some improvements.

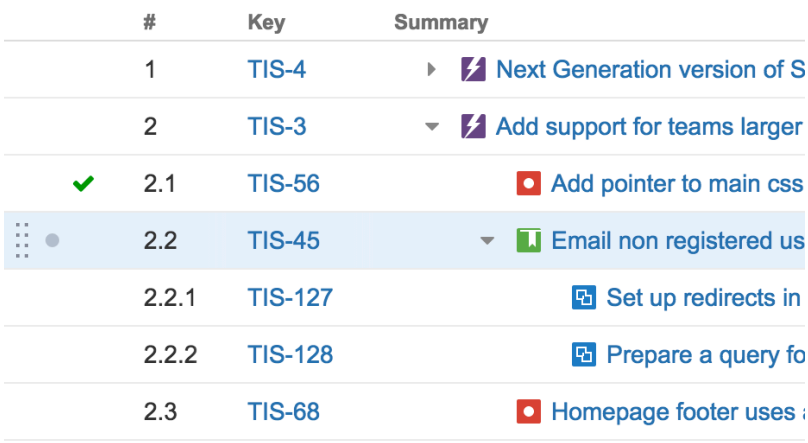
- When you open Structure tab on the project page, the displayed structure is automatically filtered to show you only the issues from the current project.
- A second panel is shown that displays all issues from the current project that are not added to the structure.

Documentation: Structure on the Project Page(see page 95)

7.35.2.3 Sequential Index Column

You can now add a column that would show you a hierarchical number (for example, "2.1.15"), based on the item's position in the hierarchy.

Documentation: Sequential Index Column(see page 899)



7.35.2.4 Quick Action Lookup

For those who love using keyboard, we added a special shortcut – **`s,q`** – that displays a lookup window where you can find all actions you can take by their name.

You can use this lookup to see if there's an action for what you need to do or to look up keyboard shortcuts for actions.

Documentation: Quick Action Lookup(see page 125)

7.35.2.5 Structure Gadget in Confluence

We have fixed several problems related to Structure's dashboard gadget being used in Confluence.

Unfortunately, a problem with loading custom fonts still remains due to an an issue in JIRA[556], so some of the custom Structure icons may not be displayed inside Confluence. There's a workaround – see Setting Up CORS Filter in JIRA(see page 1011).

Documentation: Using Structure Gadget in Confluence(see page 899)

556 https://jira.atlassian.com/browse/JRA-61400

7.35.2.6 Notable Fixes and Other Improvements

The following issues have been addressed in Structure 3.2:

- Fixed: On JIRA Data Center, Health Check reports "com.almworks.jira.structure.autosync" cluster lock problem.
- Fixed: JIRA start may take a long time due to a cluster lock or database initialization issue.
- Fixed: Excessive warning messages in log files during reindex.
- Fixed: Performance issues for large structures.

7.35.3 Supported Versions

Structure 3.2 and all extensions support JIRA versions from 6.3 to 7.1. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

Structure 3.2 is the last minor version of Structure 3 to support JIRA 6.3. Structure 3.3 will most likely support JIRA versions starting from 6.4.

7.35.3.1 Structure.Pages Upgrade Required

Please note that if you're using Structure.Pages, you will have to upgrade it to version 1.0.1.

7.35.3.2 Structure.Testy Upgrade Required

Please note that if you're using Structure.Testy, you will have to upgrade it to version 2.0.2.

7.35.4 Installation and Upgrade

7.35.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.35.4.2 Upgrading Structure

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.x.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Upgrade Structure.Testy[557] and Structure.Pages[558] add-ons if you're using them.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.35.5 Structure 3.2.1 Release Notes

5th of July 2016

Structure 3.2.1 is a bugfix release based on Structure 3.2.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[559]

7.35.5.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has changed in Structure 3!

This is a patch release based on Structure 3.2. We have addressed the following issues:

- Fixed: cannot edit generators and folders after restore.
- Fixed: impossible to add new items after restore (Oracle, Postgres).
- Fixed: default view settings are not restored.
- Fixed: link extender doesn't react to new issues with links.
- Fixed: groupers with a level setting react incorrectly to generator and structure changes.
- Fixed: filters with a level setting remove everything on lower levels.
- Improvement: optimize synchronizer audit log table.

Upgrade is recommended to all Structure users.

557 https://marketplace.atlassian.com/plugins/com.almworks.testy
558 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
559 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.35.5.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.35.5.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.35.6 Structure 3.2.2 Release Notes

30th of January 2017

Structure 3.2.2 is a security patch release for JIRA 6.3.x.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[560]

7.35.6.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has changed in Structure 3!

This is a patch release based on Structure 3.2.1. We have addressed two medium-to-critical security issues, affecting all Structure versions starting with 3.0. Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

560 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is required for all JIRA instances running Structure versions 3.0–3.2.1 and using JIRA 6.3.x.

If you're using JIRA 6.4.x., please upgrade to Structure 3.3.4(see page 896) or later version of Structure 3.3.x series.

If you're using JIRA 7.x, please upgrade to Structure 3.5(see page 874) or later.

> ⚠ You should have Structure license with active maintenance (expiring not earlier than January 30th, 2017) to run the patches.

7.35.6.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.35.6.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
> - If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.35.7 Structure 3.2.3 Release Notes

> ⓘ **19th of June 2017**
>
> Structure 3.2.3 is a security patch release for JIRA 6.3.x.

Download from Archive(see page 1165)
Structure on the Atlassian Marketplace[561]

561 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.35.7.1 Patch Release

This is a patch release based on Structure 3.2.2. We have addressed a critical security issue, affecting all Structure versions starting with 3.0. Details of the problem are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Upgrade is required for all JIRA instances running Structure versions 3.0–3.2.2 and using JIRA 6.3.x.

If you're using JIRA 6.4.x., please upgrade to Structure 3.3.5(see page 897) or later version of Structure 3.3.x series.

If you're using JIRA 7.0.x., please upgrade to Structure 3.5.1(see page 880) or later version of Structure 3.5.x series.

If you're using JIRA 7.1 or later, please upgrade to Structure 4.1(see page 856) or later.

You should have Structure license with active maintenance (expiring not earlier than June 20th, 2017) to run the patches.

7.35.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.35.7.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.36 Structure 3.1 Release Notes

30th of April, 2016

Structure 3.1 adds support for the upcoming Structure.Pages extension and contains a few fixes and improvements.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 909) before upgrading.

Download Structure 3.0 and Extensions
(see page 1164)Structure 3.0 Demo Server[562]
Structure 3 Quick Start Guide
(see page 906)

7.36.1 Incremental Update

Structure 3.1 is an incremental update in the Structure 3 series. It contains a number of fixes and improvements, and also adds support for the upcoming Structure.Pages release. There are no new features in this version.

Upgrade is recommended; it is required if you plan to install Structure.Pages or if you experience any issues with the current version of Structure.

7.36.1.1 Structure.Pages Release

Structure.Pages is our newest extension for Structure, which adds integration with Confluence and support for Confluence pages as another type of items in the structures. Check out Structure.Pages 1.0 Release Notes[563] for more details.

As this is the first public release of Structure.Pages, it will be some time until Structure.Pages appears at the Atlassian Marketplace. Until then, you can download Structure.Pages from the Download(see page 1164) page.

7.36.1.2 Notable Fixes and Improvements

The following issues have been addressed in Structure 3.1:

- Added support for sorting by Script Runner's numeric custom fields.
- Improved Structure database lock-out during restore from backup.
- Fixed: Cannot upload changes after new issue creation fails.
- Fixed: Field column does not use proper number formatting for totals.
- Fixed: Excessive warnings from synchronization engine during restore from backup.
- Fixed: Changing direction in sorting by resolution doesn't work.
- Fixed: Error 500: "failed to upload structure changes" appears on creating issue under unavailable item

562 https://demo-structure3.almworks.com/
563 https://wiki.almworks.com/display/pagesmaster/Structure.Pages+1.0+Release+Notes

7.36.2 Supported Versions

Structure 3.1 and all extensions support JIRA versions from 6.3 to 7.1. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

⚠ Structure 3.1 is the last minor version of Structure 3 to support JIRA 6.3. Structure 3.2 will most likely support JIRA versions starting from 6.4.

7.36.2.1 Structure.Testy Upgrade Required

Please note that if you're using Structure.Testy, you will have to upgrade it to version 2.0.1.

7.36.3 Installation and Upgrade

7.36.3.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.36.3.2 Upgrading Structure

⚠
- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Upgrade Structure.Testy[564] add-on if you're using it.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

564 https://marketplace.atlassian.com/plugins/com.almworks.testy

7.36.4 Structure 3.1.1 Release Notes

2nd of June 2016

Structure 3.1.1 is a bugfix release based on Structure 3.1.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[565]

7.36.4.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has changed in Structure 3!

This is a first patch release based on Structure 3.1. We have addressed some problems that were reported by our customers and made some improvements. Upgrade is recommended.

Among the issues addressed in Structure 3.1.1:

- Fixed: infinite generator creation in case of complicated structure.
- Fixed: a deadlock caused by JQL generators using the structure() function.
- Fixed: big syncronizer audit log retrieval.
- Implemented: synchronizer audit log cleanup.

7.36.4.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.36.4.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

565 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.37 Structure 3.0 Release Notes

5th of April, 2016

Structure 3.0 is the biggest, most extensive and most awaited update of the Structure add-on. It introduces a lot of amazing features that can help you bring your command of JIRA to the next level.

Important! Upgrade from Structure 2.x requires additional manual operations.

This version is a massive overhaul of Structure. If you're using Structure 2, please pay attention to these Release Notes, especially to Upgrade Instructions.

Download Structure 3.0 and Extensions
(see page 1164)Structure 3.0 Demo Server[566]
Structure 3 Quick Start Guide
(see page 909)

7.37.1 Structure 3 – a Different Experience

Structure 3.0 is very different from previous versions of Structure. While the main concept – "structure" as a hierarchical list of things – remains, many things changed and a lot of features were added. We have put a lot of thought into expanding the scope of the product and rebuilt Structure almost from ground up.

After upgrading and finding your way around new user interface, you should be able to work with Structure 3.0 in the same way you worked with Structure 2.x. However, to take advantage of the new features such as Automation, some learning will be required. When you have Structure 3 installed, check out menu **Structure | Get Started** for a crash course on most important features. The Structure 3 Quick Start Guide(see page 909) has more details and is a recommended reading.

7.37.2 Structure 3 Highlights

- Multi-parent (same issue can be at multiple locations in any structure)
- Folders (special non-JIRA folders for containing issues in structures)
- Issue sorting and grouping
- Structures within structures
- Automation (automatic structure generation, an alternative to synchronizers)
- Revamped user interface
- JIRA Data Center compatible

566 https://demo-structure3.almworks.com

Full description of all new features would be too much for this document. We invite you to take a look at the Structure 3 Quick Start Guide(see page 909) or explore the add-on yourself (make sure you stop by **Structure | Get Started** menu).

7.37.3 Notes on Structure 2 Features

⚠ Please read carefully: if you're relying on some of the changed features, you need to make sure that after upgrade everything works as expected.

7.37.3.1 Synchronizers

Synchronizers were remade. They generally do the same things as before, but we upgraded their logic so they can take advantage of the new architecture. For example, Links synchronizer can now place an issue in two or more places in the structure, so you will be able to visualize non-hierarchical links structure.

We kept most options, but introduced "Source of Truth", which helps bi-directional synchronizers resolve conflicts. If you're using synchronizers, please revisit all synchronization settings after upgrade.

ⓘ **Synchronizers and Generators**

Please note that generators "see" the results of synchronizers' work, but not vice versa. if you'd like to take advantage of the new feature, Generators(see page 140), and you're using synchronizers, we advise you to start off with a new structure to avoid confusion.

7.37.3.2 Other Changes

- Now an issue may be in several places in a structure. If you're using columns that calculate totals, the columns now have "Exclude duplicates" option, which makes sure that each issue is counted only once.
- Structure 3.0 does not have separate backup and restore, and there's no migration feature yet too. We'll reinstate these features in one of the upcoming versions.
- We changed the way Structure treats issues from projects that are not enabled for Structure. Such issues still won't be shown as a part of search result, however, if such an issue happens to be in a structure, it won't be hidden.
- If a user does not have access to issue A, but has access to its sub-issue B, the user will see "Unavailable item" instead of issue A. (In Structure 2, sub-issue B was elevated up one level to replace A in this case.)

7.37.4 Supported Versions

Structure 3.0 and all extensions support JIRA versions from 6.3 to 7.1. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

7.37.4.1 Browser Support

We now have only partial support for Internet Explorer 9 and 10. There are some known issues, but you may be able to use Structure. If you encounter a problem while using IE9 or IE10, please let us know and we'll advise if it's a known problem or something we can deal with.

If you're using Structure.Testy, you need to install Structure.Testy 2.0. Earlier versions of Testy are not supported.

7.37.5 Compatibility Issues

7.37.5.1 Other JIRA Plugins

If you're using a plugin that integrates with Structure 2.x, most likely it will not work with Structure 3.0. Here's the suggested course of action:

- Structure.Testy – we are releasing Structure.Testy 2.0, which is compatible with Structure 3.0. If you're using Testy, please upgrade Testy to the new version as well.
- `scheduled-sync`, `status-bar-column` or other complimentary plugins by ALM Works, published as a part of Structure 2.x documentation – these plugins will not work with Structure 3.0. We will release updated versions later, along with the Structure 3.0 final release. If you need one of those plugins urgently to try out Structure 3, please contact us.
- Custom plugins that integrate with Structure, whether developed by ALM Works or an independent vendor – these plugins will most likely not work with Structure 3.0 and will require additional work to be made compatible. For plugins made by ALM Works, please contact us with an upgrade request. If you have your own integrations or a plugin made by an independent contractor, you'll need information about the new APIs.
- Gantt-Chart for JIRA – if you're using this plugin and use its integration with Structure, you might want to hold off the upgrade until a compatible version is available.

7.37.5.2 REST API

REST APIs also have changed in Structure 3.0. If you have any reporting scripts that connect to Structure via REST, they most likely will stop working.

7.37.5.3 Remote Gadget Not Available

If you're using Structure dashboard gadget in Confluence, it is currently known to have issues. We'll be reviewing this functionality and possibly providing alternatives after Structure 3.0 release.

7.37.6 Changes in API

In Structure 3.0 we have made massive changes in the architecture of the product to accommodate the new features and lay groundwork for future expansion of the platform. Unfortunately, that means a lot of incompatible changes in the API. Most integrations with Structure 2.x will not work with Structure 3.0.

The new Java and REST API will be published later, closer to Structure 3.1 release. The reason for that is that we'll need to spend additional time after Structure 3.0 release to stabilize and clean the new APIs and provide sufficient documentation.

Until the new stable API is released, we can provide information about the new API on individual basis.

7.37.7 Installation and Upgrade

> **Important:** The data from Structure 2.x is not automatically transferred when you upgrade, so you'll need to manually migrate it after installation.

7.37.7.1 Installing Structure for the first time

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure 3.0 add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Startedto visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.37.7.2 Upgrading from Structure 2.9–2.11

> If you have Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

1. Create backup of current Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the whole `structure/` sub-directory under JIRA home while Structure plugin is disabled.
2. Download and install Structure 3.0 add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
3. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
4. To transfer the data, open **Administration | Structure | Export Structure 2.x Data** page. It allows you to create a backup zip with all Structure 2.x data, and then opens **Restore Structure** page, allowing you to immediately import the backup into Structure 3.x database.

   > **Important**: after you press **Create Backup**, wait for the backup to finish – it might take a while! Do not try to go back and repeat the backup. To see if the backup is still going on, access JIRA home directory, `export/` folder, and see if the backup file is growing in size.

   a. Alternatively, you can use **Administration | Structure | Restore Structure** menu and use any Structure 2.x backup made earlier.
5. If you have Structure.Testy installed, upgrade to Structure.Testy version 2.0 or later.
6. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

> Unlike previous versions, Structure 3.0 uses the main JIRA database to store its data. After upgrade, your Structure 3.0 database will be empty. Don't panic – all data is intact, but it must be transferred from Structure 2.x database. See the instructions above.

7.37.7.3 Upgrading from preview versions of Structure 3.0

If you have installed Structure 3 beta or release candidate, you need to uninstall it first.

1. Download and install Structure 3.0 add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version! All data will be safe.
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. If you have Structure.Testy installed, upgrade to Structure.Testy version 2.0 or later.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.37.7.4 Upgrading "Global Structure"

If you're using "Global Structure" structure, which was created by default in Structure 2.x, you need to make sure that there's an "owner" of that structure. Otherwise, Automation feature will not work there.

1. Open **Structure | Manage Structure**.
2. Find Global Structure and check if it has non-empty Owner.
3. If it doesn't have an owner, click **Configure**, and set yourself as the owner.

7.37.7.5 Downgrading

If you decide to downgrade to Structure 2.11, you can do so, but any changes made in Structure 3.0 will not be transferred to the previous version.

7.37.8 Structure 3.0.1 Release Notes

> **11th of April 2016**
>
> Structure 3.0.1 has critical fixes related to SQL Server database, upgrade from Structure 2, performance issues and compatibility with JIRA 7.1.4.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[567]

7.37.8.1 Patch Release

> If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has changed in Structure 3!

This is a first patch release based on Structure 3.0. We have addressed some critical problems that have surfaced after the initial Structure 3. Upgrade is highly recommended.

567 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

SQL Server Issues

There were a number of problems that appeared on some JIRA instances that were running on SQL Server database. Most prominently, there was a chance that the add-on would not work with a "*Invalid object name*" message.

Additionally, there were potential problems and database connection leaks when restoring Structure data from backup. All these are fixed in Structure 3.0.1.

> If you have been using Structure 3.0 and have migrated your data from Structure 2, please check your Database Monitoring page when there's no activity in JIRA. If you see a constant number of active connections in the pool when there's no work load, consider restarting JIRA.

JIRA 7.1.4 Compatibility

Some backwards-incompatible changes were introduced in Atlassian JIRA 7.1.4, resulting in the layout switch not working.

Structure 3.0.1 is fully compatible with JIRA 6.3 – 7.1.4.

Improved Migration

Unfortunately, we didn't attract enough attention to the fact that when you upgrade to Structure 3 from Structure 2, you need to re-import all of your Structure data. This introduced some confusion, as the structures appeared to be lost.

To improve the migration process, we now offer JIRA administrator to upgrade Structure data immediately, using a top-of-the-screen banner.

Additionally, the backup and restore processes now happen in the background, so the browsers won't time out when the procedure takes longer than usual.

> **Important**! If you already have migrated your Structure 2.x data, just close the banner that suggests you to do the migration (again). If you restore Structure 2.x data once more, you will roll back your Structure data to the state before the upgrade.

Miscellaneous Fixes and Improvements

- Removed status icons on JIRA 7+
- Performance optimizations
- Smaller fixes and improvements

7.37.8.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.

3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.37.8.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from version 3.0.0 is simple:

1. Create backup of JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data.
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.37.9 Structure 3.0.2 Release Notes

26th of April 2016

Structure 3.0.2 is a bugfix release based on Structure 3.0.

Download the Latest Version(see page 1164)
Structure on the Atlassian Marketplace[568]

7.37.9.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 909) – a lot has changed in Structure 3!

This is a second patch release based on Structure 3.0. We have addressed some problems that were reported by our customers and made some improvements. Upgrade is recommended.

Among the issues addressed in Structure 3.0.2:

- Fixed: synchronizers cycle protection fails to send e-mails to structure owners.
- Fixed: synchronizers cycle protection causes synchronizers become instantly disabled if a cycle happens at least once.
- Fixed: problems with reassigning issues via drag-and-drop when the assignee has their login name changed in the past, or when the user is imported from an LDAP directory. (Also affects other fields that refer to users.)
- Fixed: structure synchronizers created in Structure plugin version 1.3 or earlier are not properly migrated during transition from Structure 2 to Structure 3.
- Fixed: printable page and Excel export do not show values for certain fields, such as Total Remaining Estimate and other duration-type fields, Total Votes and various configurations of the Progress field.

568 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

- Fixed: error when trying to create a new structure with Agile template, but the Agile board does not have Rank ordering.
- Fixed: printable page and Excel export disregard "Exclude Duplicates" setting for the column.

7.37.9.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 1164) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.37.9.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 909).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

8 Additional Resources

8.1 Structure Roadmap

Updated On	24 Apr 2023
Next Update	July 2023

In this roadmap, we'd like to share some of the features the Structure team is going to work on in the near to mid term. The scope of the roadmap is 1 to 2 years.

A few notes and disclaimers about the roadmap:

- We only list **new, important functionality** – we are also going to work on other things, such as improving existing features, improving quality, improving user interface, adding minor features.
- This document lists only upcoming features in Structure. We're also working on Structure extensions, such as Structure.Gantt, which have their own roadmaps.
- The roadmap is subject to change. We will update it periodically so it reflects our current vision.

⚠ It is our general approach to focus on the quality of the product. Sometimes this means delivering a product later or changing plans and priorities, as unexpected dependencies and challenges appear. Therefore, while we try to adhere to the announced roadmap, by no means should it be considered an obligation, and it should not be relied upon when making purchasing decisions.

8.1.1 Versions and Dates

We generally aim to release a minor version of Structure every 2-3 months and a major version every year. The following is an approximate release schedule for the scope of this roadmap.

Target Month	Jun'23	Sep'23	Nov'23	Feb'24	Apr'24
Version	9.0	9.1	9.2	9.3	10.0

8.1.2 Roadmap

8.1.2.1 Recently Released:

- **Structure 7.0: Formula language improvements** - several improvements to formula language, including:

    - additional functions for working with dates
    - support for arrays
    - statistical functions
    - support for fixVersion.releaseDate.min()
- **Structure 7.4: History support in formulas -** ability to use dates from Jira events, such as the date of a status change or the date the issue was added to a sprint
- **Structure 8.0: UI/UX improvements** - ability to see more data in Structure with the Text Wrapping feature
- **Structure 8.2: Saved Columns and Filter by field:** ability to share formulas inside the organization and create filters in few clicks, without JQL knowledge

8.1.2.2 Here's the list of major features that we're planning to work on next year:

- **More Notes** - ability to add multiple Notes columns
- **Planning tasks** - artificial tasks to do "sandbox planning" in Structure and then save the data to Jira
- **Structure Templates improvements** – revamp existing templates and add more
- **More integrations with other plugins(Tempo Planer, Zephyr, Insight)** - ability to use additional plugin fields for sorting, grouping, and formulas
- **Scheduled jobs (scheduled export to excel)** - ability to schedule Excel exports and have them automatically distributed via email
- **Structure Health** - ability to check if there are any problems with structures, and highlight active structures

8.2 Feature Comparison - Data Center and Cloud

The following chart outlines the key features available in Structure for Jira Data Center/Server and Structure for Jira Cloud.

✅ – Available ⚠️ – Partially available ❌ – Currently unavailable

Feature	DC/Server	Cloud
Portfolio management	✅	✅
Custom hierarchies	✅	✅
Real-time updates with Automation/ Generators	✅	✅
Configurable columns	✅	✅
Aggregate values	✅	✅
Progress reporting(see page 472)	✅	✅
Folders(see page 104)	✅	✅
Memos(see page 105)	✅	❌

Issue management		
In-line editing within Structure	✓	✓
Bulk issue editing	✓	✗
Create new issues from Structure	✓	✓
Formulas		
Formula columns(see page 463)	✓	✓ Availble with some differences - see Formulas Comparision(see page 427)
Use formulas in generators	✓	✗
Conditional formatting	✓	✓
Bundled formulas(see page 257)	✓	✓
Saved formulas (Saved columns(see page 524))	✓	✗
S-JQL(see page 428)	✓	✗
Generators / Automation		
Insert issues based on a JQL or text query	✓	✓
Insert issues based on Agile boards	✓	✓
Insert other structures	✓	✗
Insert issues based on values in Jira fields	✗	✓
Extend the hierarchy based on issue links, epics, and sub-tasks	✓	✓
Group by issue field	✓	✓

Sort by issue field	✅	✅
Filter based on a JQL query	✅	✅
Manual adjustments(see page 194) to automated structures	✅	❌
Templates	✅	✅ Reimagined in Cloud as Presets[569]
Effectors(see page 179)	✅	❌
Transformations(see page 215)	✅	⚠️ Only filters (see page 207)are available at this time
Structure columns		
Progress columns(see page 472)	✅	✅
Notes column(see page 470)	✅	❌
Time in Status column(see page 493)	✅	✅
Aggregate columns(see page 497)	✅	✅
Tempo Work Logged column(see page 491)	⚠️ Work Logged column(see page 499) currently does not support billable time	✅
Test Run column[570]	⚠️ Requires the free Structure.Testy extension	✅
Horizontal scrolling(see page 503)	✅	✅
Pinned columns	✅	✅

569 https://wiki.almworks.com/display/structure/.Presets+v7.4
570 https://wiki.almworks.com/display/structure/.Test+Run+Column+v8.0

Text wrapping(see page 523)	✓	✗
Sort by columns	✓	✓
Structure management		
Structure sharing	✓	✓
View sharing	✓	✓
Change history	✓	✗
Undo	✓	✓
Permissions		
Define permissions based on groups, users, and project roles	✓	✓
Enable/disable Structure for specific groups	✓	✓

Migrating to Jira Cloud or Jira Data Center

If you're currently using Structure on Jira Server, learn more about your options:

- Migrating to Cloud(see page 930)
- Migrating to Data Center(see page 940)

8.3 Structure Best Practices

Because all organizations are different, and teams within those organizations are different still, Structure was designed to be versatile and flexible enough to handle a diverse range of unique use cases. Therefore, there is no single, best strategy to rolling Structure out across an organization. However, there are best practices which can be utilized to reduce disruption and maintain peak performance levels.

We will review these best practices, collected from our experience as well as customer feedback. It is important to understand that the best practices referenced here were developed with optimal system performance at the core.

Important information regarding Performance Considerations can be found below.

8.3.1 Initial Considerations

8.3.1.1 Organizations New to Jira

If you have Structure as part of your Jira solution and your organization is still new to Jira, it may be best to consider the solution as a whole. That is to say, include it in your basic Jira onboarding process. A lot of our training material and resources are well suited to this. There are plenty of solutions consultants who can help with training as well.

8.3.1.2 Organizations Established in Jira

It is highly recommended to proceed with an incremental rollout in this situation. By utilizing Structure configurations, you can select specific groups and/or projects that will have access to Structure initially. You can then expand the access as your teams become more familiar. To add an additional layer of control to the roll-out, you can also limit the type of access each user or role has to read or edit structures. Typically, the first groups should be your champions - those who may have recommended Structure in the first place and whom you will rely on to guide additional users through best practices.

For detailed instruction on how to adjust permissions, read Enabling Structure(see page 921)

8.3.2 Basic Terms & Principles

8.3.2.1 Automation

Automation uses generators, i.e. special rules, that tell a structure what issues to show you from Jira and where to place them within the structure. Whenever a structure is open, these generators are running and rebuilding your structure as changes are happening to the associated Jira data. In this way, you know that your structure is always up-to-date and relevant

To learn more about Automation a good how to video can be found here: Generators(see page 140)

Generators

- Insert - pulls in an initial set of issues, from which the rest of the structure is built
- Filter - hides some issues, or certain levels, from the structure
- Sort - reorders issues by the values in a specified field
- Group - organizes issues under the values of a field
- Extend - adds additional issues based on links to the issues already in the structure

Generators can only be added under a “static” part of the structure. In other words, anything that was not populated by another generator. Below are the two most common locations & their use case.

1. At the top of the structure - this will apply the generator to the entire structure. This is the best option if you need to organize the entire structure in the same way.

2. Under a folder or issue - this limits the generator to just the items beneath that folder or issue. This allows you to combine several sets of issues in one structure and organize them differently by adding different generators under each folder.

> Learn more about how to set up Generators here: Adding a Generator(see page 141)
>
> Reminder: In order to add additional generators, you must first select the top line of the structure (to apply it to the entire structure) or the folder or static issue (to apply it to only a part of the structure).

8.3.3 Things to Keep in Mind

Before we get started setting up a structure, there are a few key items to keep in mind as you get going. These are important to maintaining desirable performance levels.

1. Extend generators should have limited levels . The default is set from 1-10. We recommend adjusting for 1-2.
   Click here to read more...

   When added to a structure, an Extend generator checks every configured level in the structure to determine whether issues at that level need to be extended. For example, a Linked Items Extender will check every issue in every configured level to see whether or not a specific link type is present, and add linked issues whenever necessary. Checking an individual issue for links doesn't take long, but checking thousands of issues may.

   When configuring extenders, we recommend limiting the scope to just those levels that need to be extended. This will prevent your structure from checking unnecessary issues and can provide a significant performance improvement for large structures.

   See Generator Scope(see page 198) for more information

2. Use Filter generators sparingly . Adjust the initial insert generator query instead.
   Click here to read more...

   It’s not only the final number of issues in a structure that is important. It's also important to check the number of issues before any filter generators are added.

   For example, if you add an Insert generator with several Extend generators, they may pull 100k issues into the structure, but then a Filter generator may remove most of those, so you only see a few thousand issues. Even though it looks like a small structure, it still has to add all those original issues (and then put extra effort in to remove most of them), which could result in very poor performance

3. Aim for multiple, specialized, structures . Avoid large “all encompassing” structures.
   Click here to read more...

   Some structures are simply too large or too complex to be loaded as fast as desired. This may be fine for infrequently-used structures, but for structures that are used every day by multiple users, we recommend either decreasing the size of the structure (see above) or splitting the large structure into two or more smaller structures.

   For example, instead of having a global structure used by all teams, create several smaller structures specific to each team for day-to-day use, and reserve the larger global structure for cross-team planning sessions.

8.3.4 Common Approaches to Building

8.3.4.1 Top Down

We recommend utilizing this approach when a full overview is needed. Specifically where it is important to roll-up data such as estimates, time spent and progress from lower levels all the way to the top. It is strongly recommended to limit access to the appropriate users, such as project and program managers. We do not recommend using this method for a team's main structure for daily work or as the default structure.

Click here to read more...

Because of the nature of the use case, such structures can get pretty big. We are visualizing the entire scope of work. It's not necessarily a problem, but these structures are usually more performance heavy than smaller structures, especially if you have many levels connected by links. It's perfectly fine to use them when a full overview is needed, but it may not be the best approach for a general-use structure utilized by many users on a daily basis.

Additionally, this type of structure should not be used as a default structure. Default structures are opened by default on the issue page, which means every time a user opens an issue, this whole structure is loaded and then filtered to only show the part of it related to the issue.

How to build:

Start by adding top-level items using an Insert generator, and then use a number of Extend generators to pull in related items. Here are the common steps:

1. Add the Insert generator that will add the top level items. The JQL Inserter is the most flexible option, but it may be desirable to add issues from a particular Agile board.
2. Add Extend generators to pull in child issues. Depending on your setup, you may have to add several extenders to visualize different types of relationships. For example, you could build a 4-level structure as follows:
    - First level - add all Initiatives using a JQL Inserter
    - Second level - add epics under those initiatives with an Issue Link Extender
    - Third level - add stories using a Stories Under Epics Extender
    - Fourth level - add sub-tasks using the Sub-tasks Extender

It is strongly recommended to set the Levels option for all Extend generators - this will make everything more consistent and help with performance (especially in larger structures). In the example above, you'd set the Levels to Current level only for the first Link Extender. To pull in stories under epics, you will set the levels option: *2 to 2*, since the Epics are on the second level, and we don't need to check if there are any stories under stories. The sub-tasks extender would then be applied to levels *3 to 3*.

8.3.4.2 Focus on a Level

This approach is best for when you want to see how the scope of issues you are responsible for fits into the full hierarchy. For instance, you want to view the stories from your sprint and visualize which Initiatives and/or Epics they fall under, as well as the sub-tasks they are comprised of.

Click here to read more...

You may be wondering why this scenario would not call for the Top Down approach as well. You are, after all, trying to view a Parent > Child > Grandchild hierarchy. The difference here is in the focus. We are primarily interested in

the Child, or middle level of the hierarchy. For performance reasons, we would not want to pull in all of the top level, Parent, issues, just to filter some or most of them out.

A better approach in this situation is to start by pulling in the scope you want to see and then use Group generators to show the levels above. This will produce a much smaller structure straight away, and Groupers are more efficient in terms of CPU usage than Extenders.

How to build:

Use an Insert generator to add the initial scope - the issues you are focused on - and then use Group generators to show the parents and Extend generators to show the children. Let's look at an example where we want to build a structure using an Initiative -> Epic -> Story -> Sub-task hierarchy, but only with stories from a particular sprint:

1. Use an Insert generator to add the initial scope. In this case, we would use the JQL Inserter to pull in issues from a certain sprint.
2. To add children (in this case sub-tasks), add a Sub-tasks Extend generator (don't forget to set the correct level settings - in this case, Current level only).
3. Next, add a Group by Epic generator to show the direct parents. The stories will be distributed between the epics they belong to. Any stories without an epic will end up in the "No Epic" folder. If the parents are connected to children with issue links instead of epic links, use *Group by Issue Links*, and select the link type and direction.
4. To include Initiatives above your epics, add a second Group generator. There are two important things you need to do here:
    a. As you add this second Group generator, make sure you select the "Consider other groups" option. Without this option, the generator will try to group not the level we've just created, but the issues that were added by the original Insert generator.
    b. Once the generator is added, reverse the order of these two Group generators in the generators list, so that the one you want to group by first is higher.
5. If there are more levels above - keep adding more Group generators, until you get the highest level you need.

8.3.4.3 Visualize the Entire Scope

When you are in a hurry, but also want to make sure you didn't leave anything behind, this approach will come in handy.

Click here to read more...

This is a very quick and easy solution. It can also be very resource intensive. We recommend utilizing it only in situations where you are limited on time and are looking to populate from a particular project, component, version, team, etc. In other words, the more specific the focus the better.

How to build:

Unlike the top-down approach we used above, where we started with just our top-level issues, in this case we want to start by inserting our entire scope of issues. Then we will use other generators to organize them into the hierarchy. Here is how to build it:

1. Use an Insert generator to pull in all the issues you want to see.
2. Add an Extend generator to pull in the child issues like we've done before. The important difference in this case is that as you add Extend generators, you will be getting duplicate issues - one instance of the child issue is added to the top level by the Insert generator, and then another is added by the Extend generator. To hide the duplicates, proceed to the next step.
3. Add the Remove Inserter/Extender Duplicates Filter generator - this will hide any issues added to the top level by the Insert generator if they were also added later by an Extend generator. Any issues that are not duplicated will remain on the top level.
4. The Extend generators we have added could pull in some issues which are outside of the original scope we defined. For example, epics in our project can have some stories from other projects too, which the Stories

Under Epics Extender would pull in. If we only want to see issues from our original scope, we'll need to add a JQL Filter generator with the same query we used originally.

⚠ It is very important to set level parameters for extend generators in this configuration - even more so than in the top-down approach. Failing to do so can seriously affect performance.

If you do not restrict the scope of Extend generators by setting correct levels, you will end up with not just 2 instances of all child issues, but with an instance per level, which can be a lot of issues if you have 4-5 levels.

8.3.5 Performance Considerations

8.3.5.1 Automation

When evaluating Automation performance, there are several factors that come into play

1. The size of structures (the number of issues in each structure).
2. The complexity and the depth of structures. The more levels a structure has, the more calculations need to be done to build it.
3. Number of concurrent users editing large structures.
4. The frequency of usage. If you have a really large structure that is used on a daily basis or is set as the default structure(see page 537), this may have a negative effect on the server performance. On the other hand, a very large structure that is only used by a small group of managers when they really need it would have minimal impact.

The number of structures with generators does not affect the performance - if a structure is not opened, its generators are not running.

ⓘ To reduce the risk of Automation affecting Jira performance, we have introduced the Automation Timeout feature. This feature stops generation if it exceeds a certain, user-defined time period. The user is then able to adjust the generator settings. See Automation Paused(see page 201) for more details.

8.3.5.2 Progress, Totals and Formula Columns

The Jira fields used in columns, by themselves, have little impact on performance. They are loaded only for the structure elements which are currently displayed, in addition to a number of items below and above (to ensure smoother scrolling).

However, for progress, totals and some formulas, the values are loaded for the entire hierarchy. This is due to the fact that such columns aggregate data across the entire hierarchy. For structures with thousands of issues, this can become noticeable. This is still far less of an impact when compared to loading the hierarchy itself.

8.3.5.3 Structure on Issue Page

Users can see structures on the main Structure Board, on Jira dashboards, the project page and on the issue page. A structure which is visible in any of those locations requires generation of the structure.

For high traffic projects, this could affect performance significantly. It is recommended to use smaller and simpler structures as your default structures. To learn more about selecting the default structure for the Issue Page, see Structure Options for the Issue Page(see page 93).

You can also remove the Structure widget from the Issue Page(see page 973) completely.

8.3.5.4 S-JQL

S-JQL(see page 428) is an extension for JQL that allows you to run structure-based queries. Every time such a query is executed, the structure that is used in the query will be loaded. If you create a saved filter with S-JQL for a large structure and use it heavily somewhere, performance may begin to suffer.

8.3.5.5 Monitoring & Troubleshooting Structure Usage

The best way to check statistics on Structure usage, see structure sizes and assess load time is by reviewing the Performance Audit Log (PAL). For detailed information on the type of data which can be found in the PAL, as well as how to interpret it, additional information can be found here: Monitoring and Troubleshooting Structure Usage(see page 942)

8.3.6 Additional Resources

- Getting Started with Structure(see page 56)
- Structure User's Guide(see page 82)
- Administrator's Guide(see page 929)
- Structure Developer's Guide(see page 1029)
- Frequently Asked Questions(see page 671)
- Structure Troubleshooting(see page 1014)

8.4 Structure Demo Server

8.4.1 Access our demo server here: Demo Server[571]

If login and password are requested, use demo/demo.

8.5 Send Feedback

8.5.1 Please tell us what you think!

If you have any questions, feature requests or anything else to say about Structure – please tell us:

- Feature and Wish List on UserVoice[572]
- Support questions to Tempo Support[573]

571 https://demo-structure3.almworks.com/secure/StructureBoard.jspa?s=201&os_username=demo&os_password=demo
572 http://structure.uservoice.com/
573 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

8.6 Other Versions

Looking for a different version of Structure? Select your platform and version number at the top of the screen.

Or select your version below:

- Structure for Jira Cloud[574]
- Structure for Jira Data Center and Server[575]
- For versions 7.2 or earlier of Structure for Jira Data Center or Server, please refer to the Version Index[576].

8.7 Structure Extensions

Learn More and Try Now

- Structure.Gantt[577] - Gantt charts and roadmaps for Jira
- Structure.Deliver[578] - Agile project forecasting
- Structure.Pages[579] - View and edit Confluence pages in Structure
- Structure.Testy[580] - Our lightweight testing extension

View Documentation

- Structure.Gantt[581]
- Structure.Deliver[582]
- Structure.Pages[583]
- Structure.Testy[584]

574 https://wiki.almworks.com/documentation/structure/latest/cloud/structure-for-jira-32222947.html
575 https://wiki.almworks.com/documentation/structure/latest/data-center-and-server/structure-for-jira-32222947.html
576 https://wiki.almworks.com/display/docs/Structure+Documentation
577 https://marketplace.atlassian.com/apps/1217809/structure-gantt-gantt-charts-roadmaps-for-jira?hosting=datacenter
578 https://marketplace.atlassian.com/apps/1224682/structure-deliver-agile-project-forecasting?hosting=datacenter
579 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=datacenter
580 https://marketplace.atlassian.com/apps/1212033/structure-testy-test-checklists?hosting=datacenter
581 https://wiki.almworks.com/documentation/gantt/latest/data-center-and-server
582 https://wiki.almworks.com/documentation/deliver
583 https://wiki.almworks.com/documentation/pages/latest
584 https://wiki.almworks.com/documentation/strtesty

9 Administrator's Guide

This section contains information for Jira administrators about installing and configuring Structure plugin.

Quick steps to get Structure working:

1. Installing Structure(see page 944)
2. Setting Up Structure License(see page 950)
3. Getting Started with Structure(see page 929)

9.1 Migrate to Cloud

9.1.1 Before You Start

- See How Structure Cloud is different from Structure for Jira Server/DC(see page 918)
- Review our Migration Notes(see page 937)
- Check our public Structure Cloud Roadmap[585]
- Contact our support team at Tempo Support[586] if you have any questions or concerns

585 https://wiki.almworks.com/display/strcloudmaster/Structure+Cloud+Roadmap
586 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

9.1.2 Migration Options

- Manually Migrating to Cloud(see page 931)
- Migrating to Cloud Using the Jira Cloud Migration Assistant(see page 932)
- FAQ - Migrating to Cloud(see page 937)

9.1.3 Manually Migrating to Cloud

In order to manually migrate structures from Jira Server or Jira Data Center, there are a few key steps to undertake.

You will want to inventory exactly what you need to migrate from your on-prem instance to cloud. This will obviously include the individual structures, but may also include other information, such as the column definitions (views) that users want to preserve, and any Structure.Gantt configurations they may use. It is highly recommended you talk to the various stakeholders of Structure to get the correct structures, views and Gantt configurations you want to preserve.

To facilitate this conversation, a Jira admin can gather some data from Structure:

1. Visit the Manage Structure screen(see page 532) and find the structures that are popular (a popular structure is one that has been favorited more than once).
2. Pull up the Performance Audit Log (PAL) (see page 942)and see which structures were most recently loaded. The PAL can also tell you who owns each structure, so you can send targeted emails asking them which structures they want to preserve in the migration.

Once you have inventoried what needs to be moved, you will need to make a strategy for accessing that information. If the on-prem instance is still available post migration, it can serve as the source of reference; however, be mindful that certain elements may become stale (like date-based references). If the on-prem instance will not be available, you should take care to preserve the information you will need:

- Copy the structure definitions from the PAL.
- Gather views information from the Manage Views menu(see page 505) → Advanced tab. Views are harder to recreate on Cloud, but having a reference of what they were will be useful.
- Create a structure backup (alongside the Jira wide one). To do this, visit the Structure administration menu.

If you are migrating Structure.Gantt as well, you should also:

- Collect any work calendars from the admin section
- Gather configuration information from the configuration menu
- Assess each Gantt chart's current state. There is currently no way to save this data (aside from pdf/svg export), because it is generated from the Jira data. Any non-Jira information, such as resource calendars, will need to be stored manually.

After all the information is ready, you will need to make sure that all the Jira projects and fields used in your selected structures, views and Gantt configurations have been migrated to your Jira Cloud instance. Without this available, some of your future structures may be calculated incorrectly or not calculated at all.

Using the collected information, you will now need to recreate your structures, views and Gantt configurations in your cloud instance. Please note that not everything can be replicated in Structure Cloud yet because some features are not available(see page 918).

9.1.3.1 Additional Resources

The following guides may be helpful when setting up your structures and Gantt charts in Structure Cloud:

- Getting Started with Structure Cloud[587]
- Generators[588]
- Columns and Views[589]
- Getting Started with Structure.Gantt[590]

9.1.4 Migrating to Cloud Using the Jira Cloud Migration Assistant

Migrating Structure data using the Jira Cloud Migration Assistant requires Structure 7.4 or later. For best results, we recommend updating to the latest version of Structure.

The easiest way to migrate Structure data from Jira Server or Data Center to Jira Cloud is by using the Jira Cloud Migration Assistant (JCMA). When you run JCMA, the structures you select will be recreated in Structure Cloud, based on:

- Manually added issues and folders
- Items added and organized through Automation
- Structure views - including the default view and associated views for each structure

Before you begin the migration process, we highly recommend reviewing our Migration Notes(see page 937).

In order to perform a migration, you must have Admin access for Jira Server/Data Center and Jira Cloud.

9.1.4.1 Choose Your Path

Migrate Everything

The easiest way to migrate your structures is to simply mark Structure as "Needed" when you run the Jira Cloud Migration Assistant[591]. This will migrate ALL of your structures to the new Jira Cloud instance. If you choose this options, keep in mind that Structure can only add issues from projects that have been migrated to cloud using the JCMA - if you're not migrating all of your Jira projects, this could cause errors for some of your migrated structures.

Migrating Specific Structures

The guide below shows you how to migrate specific structures to your new Jira Cloud instance. This is the recommended path because it gives you more control over the migration and helps ensure the structures you do migrate have associated projects in the new Jira.

Don't do both! If you've already conducted a Jira migration using JCMA, and you marked Structure as Needed, all of your structures should have been migrated. Going through the steps below will result in duplicate structures.

587 https://wiki.almworks.com/display/strcloudmaster/Getting+Started+with+Structure+Cloud
588 https://wiki.almworks.com/display/strcloudmaster/Generators
589 https://wiki.almworks.com/display/strcloudmaster/Columns+and+Views
590 https://wiki.almworks.com/display/cloudgantt/Getting+Started+with+Structure.Gantt
591 https://marketplace.atlassian.com/apps/1222010/jira-cloud-migration-assistant

9.1.4.2 Migration Steps in Jira Server or Data Center

Step 1 - Make sure you have the appropriate software installed

The following programs must be installed on Jira Server/Data Center:

- Structure for Jira
- Jira Cloud Migration Assistant[592]

Learn more: Use the Jira Cloud Migration Assistant to Migrate[593]

Step 2 - Create a new Jira migration

Go to **Administration | System | Migrate to cloud** and create a new migration. For step-by-step instructions, see Use the Jira Cloud Migration Assistant to migrate from server to cloud[594].

Save the migration set. **DO NOT RUN THE MIGRATION SET!** Before running the migration set, you need to add your structures.

Step 3 - Create a Structure migration configuration

Go to **Administration | Structure | Migrate to Cloud** and create a new configuration. Give it the **exact same name** you used for the Jira migration in Step 2.

If the names do not match exactly, Structure data will not be migrated.

Structure can only add issues from projects that have been migrated to cloud using the JCMA. We highly recommend that you only add structures to a migration configuration if 1) the associated projects are also being migrated in that set, or 2) those projects have been migrated previously using the JCMA.

Step 4 - Run the Jira migration

Return to **Administration | System | Migrate to cloud** and run the JCMA migration.

9.1.4.3 Migration Steps in Jira Cloud

Once the migration has been run, the following steps should be taken in Jira Cloud.

Step 5 - Review the Jira migration

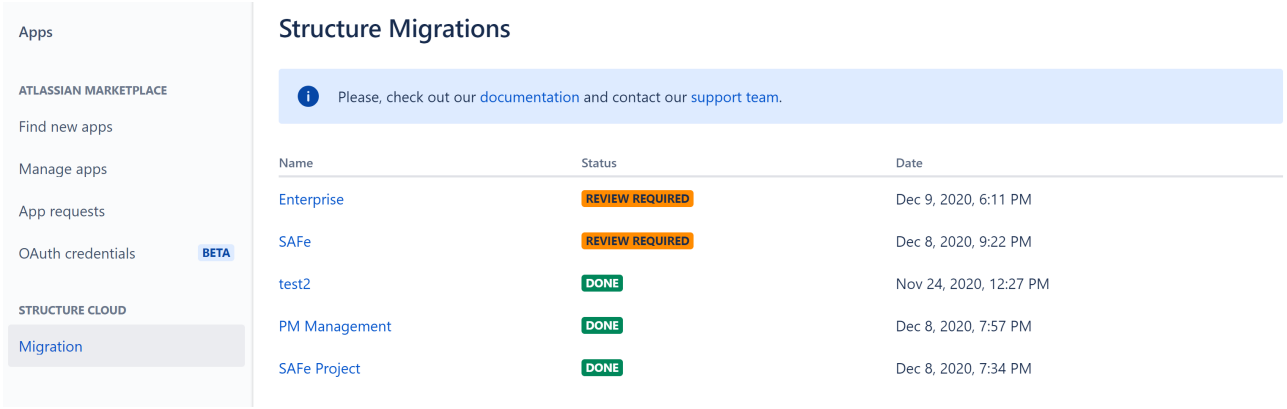
Check that all the users, groups, and projects were migrated successfully. See Use the Jira Cloud Migration Assistant to migrate from server to cloud[595] for more details.

---

592 https://marketplace.atlassian.com/apps/1222010/jira-cloud-migration-assistant
593 https://support.atlassian.com/migration/docs/use-the-jira-cloud-migration-assistant-to-migrate/
594 https://confluence.atlassian.com/cloud/use-the-jira-cloud-migration-assistant-to-migrate-from-server-to-cloud-993925215.html
595 https://confluence.atlassian.com/cloud/use-the-jira-cloud-migration-assistant-to-migrate-from-server-to-cloud-993925215.html#After%20migrating

Step 6 - Check Structure migration status

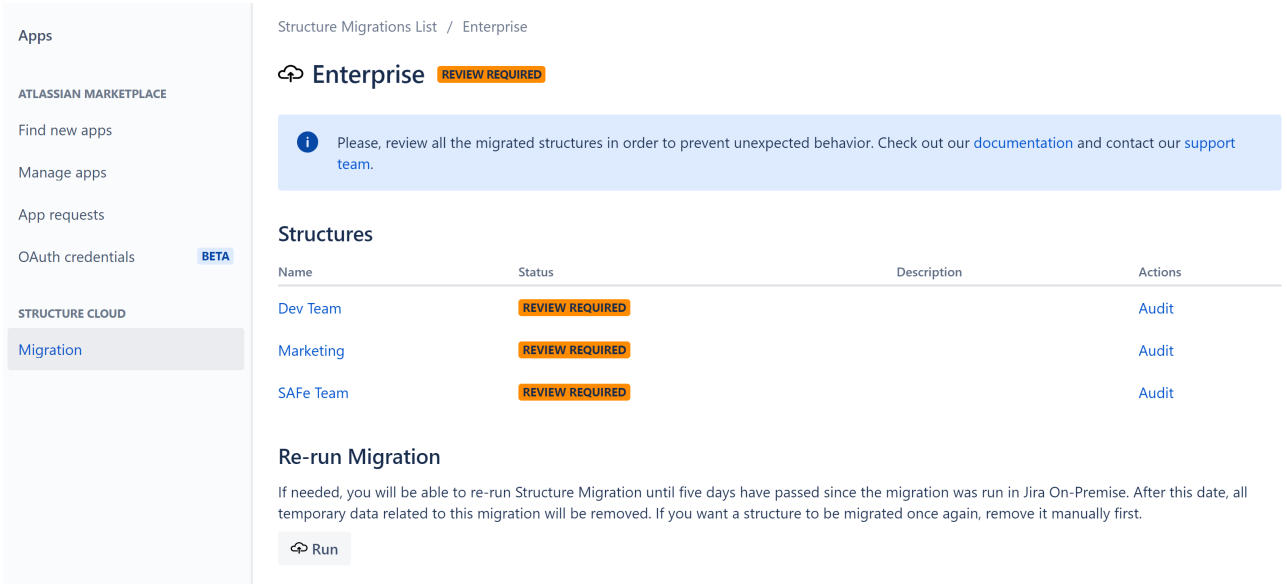
Go to **Settings | Apps**. In the left panel, locate the Structure Cloud section, and click **Migration**. This will bring up a list of migrations with the status of each.



You may receive the following statuses:

- Review required - the migration has completed, but some structures need to be reviewed
- Done - the migration has completed, and the structures are ready to be used
- In progress
- Failed

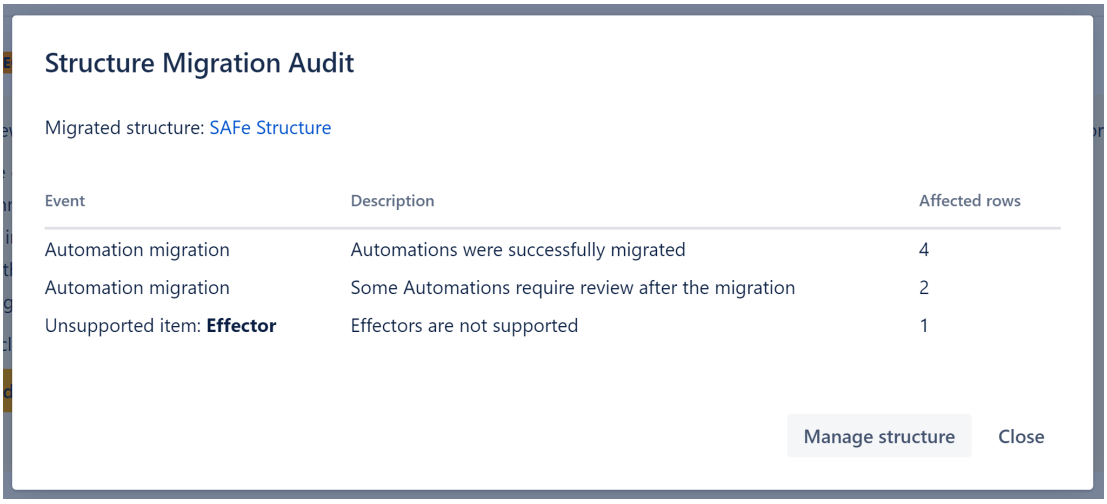
To review the structures in each migration, click the migration name.



Each structure will have one of the following statuses:

- Review required - the structure has been migrated, but needs to be reviewed
- Done - the structure has been migrated and is ready to be used
- Failed

For additional details about the migration, including which items require review, click **Audit**.



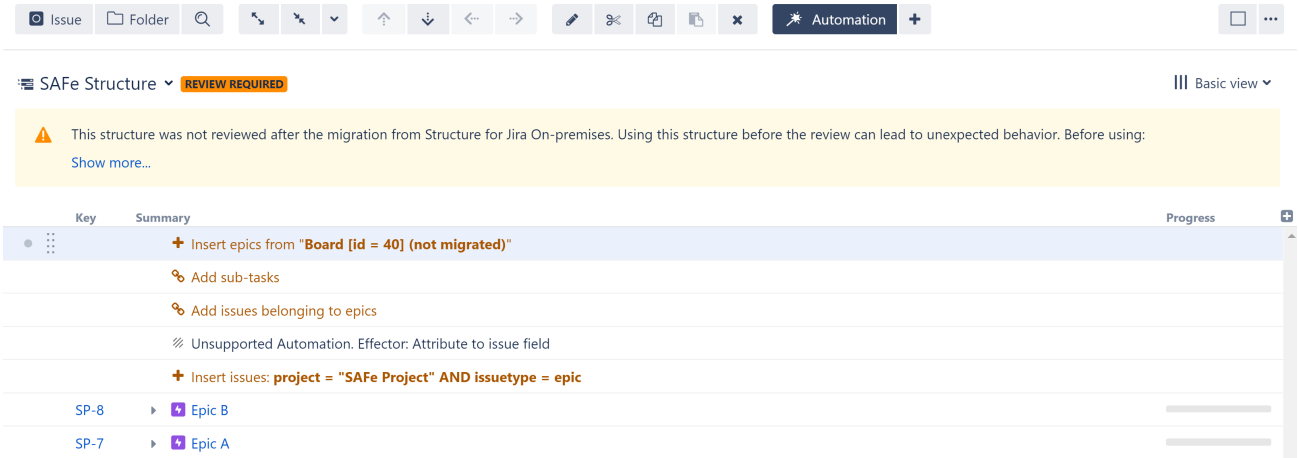
> **Re-run a migration**
>
> If you need to re-run the migration for any reason, you can do so within 5 days of the original migration. Click the **Run** icon at the bottom of the Migration screen.

Step 7 - Review each structure

To review each structure, click the **Manage structure** link on the Audit screen or the structure's name on the Migration screen.

> Before you can open a structure, you may need to update its Owner and Permissions settings via Manage Structures[596].

If there are any items that need review, you will see a message at the top of the structure with steps that should be completed before using the structure.



You should check:

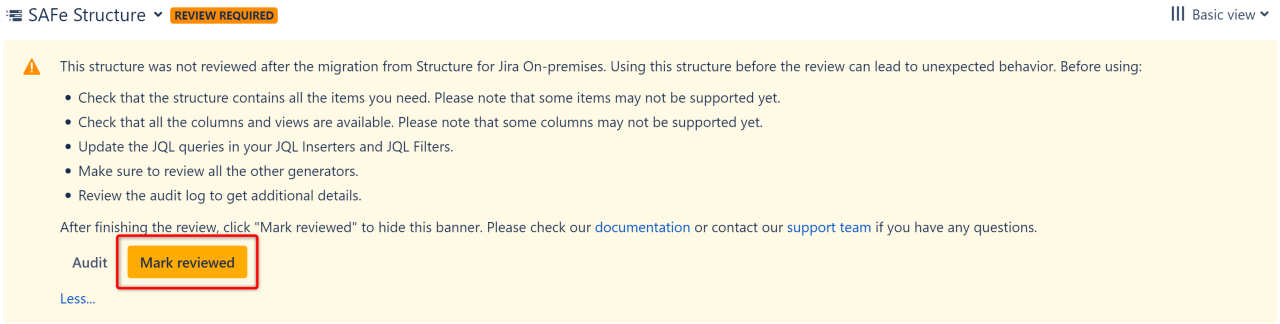
596 https://wiki.almworks.com/display/strcloudmaster/Managing+Structures

- Manually-added items - make sure they were added correctly to the structure. Note that Memos are not yet supported in Structure Cloud and will be replaced by folders.
- Automations - make sure they were migrated correctly and return the expected results.
- Automations labeled “not migrated” - this indicates structure was unable to reproduce the automation in Cloud, most likely because it could not locate a project, board, or issue. Double-click the automation and check it settings. Adjust as necessary.
- Agile Inserter - structure built using the Agile Inserter will need to be recreated in the new cloud instance.
- JQL Inserters, JQL Filters, and JQL Quick Filters – doublecheck the JQL and results, as some attributes may be different in Cloud. Adjust as necessary.
- Columns - make sure columns were migrated as expected.
- Views - only custom views associated with the selected structures will be migrated.
- Anything labeled "Unsupported" - this indicates that the item or automation is not yet supported in Structure Cloud and could not be migrated.
- Permissions – structure and view permissions are not migrated to cloud, because Server/DC and cloud use different permission models. To configure structure permissions, locate the structure in Manage Structures(see page 532), click **Configure**, and update the permission settings.

⚠ Test Run columns are not yet supported with the migration assistant.

Step 8 - Mark the structure reviewed

Once you have reviewed the structure and updated it as necessary, return to the banner at the top and click **Mark reviewed**. The banner will not appear for this structure again.



9.1.4.4 Additional Resources

- How Structure Cloud is different from Structure for Jira Server/DC(see page 918)
- Structure Cloud Roadmap[597]

✓ Need help or have questions? Contact Tempo Support[598].

597 https://wiki.almworks.com/documentation/structure/latest/cloud/structure-roadmap-32228104.html
598 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

9.1.4.5 Migration Notes

Before Migrating

Before migrating to Cloud, there are a few things you should check.

- There are some differences between the features available on Server/Data Center and Cloud. We strongly encourage you to review our Feature Comparison(see page 918) and Formula Comparison(see page 427) charts.
- Large structures - There is a 10,000 issues limit for any structure on Cloud. If you currently have structures with more than 10,000 issues, you should optimize those before updating: adjust the generators to include fewer issues, split them into multiple smaller structures, etc.
- Manual adjustments - Manual adjustments(see page 194) are not supported in Cloud. The structures will still migrate, but any hierarchies you've created using manual adjustments will be removed and appear as a flat list of issues (or the way they originally looked before manual adjustments were applied).

After Migrating

After migrating, please be aware of the following:

- Structure permissions (view, edit, automate, and control) and view permissions are handled differently in cloud, so you will need to reconfigure these after the migration. To configure a structure's permissions, locate the structure on the All Structures(see page 532) page, click **Configure**, and update the permission settings.
- It's normal for your Structure Selector menu to be empty immediately after migrating. This is usually because there are no recently used structures and/or permissions have not been configured for the migrated structures. Jira admins can view the migrated structures and assign permissions on the All Structures(see page 532) page.

Additionally, we highly recommend you review each migrated structure following Step 7 in our Migrating to Cloud(see page 932) guide.

9.1.5 FAQ - Migrating to Cloud

9.1.5.1 How long does a migration from Jira Server to Jira Cloud take?

It depends.

Factors like the size of your Server instances, how many Jira third-party apps you need to move, and the human resources available to you – whether it's in-house help from your Jira admins or Atlassian Marketplace Solution Partners – are straightforward, but others may not be so obvious upfront. Perhaps the Jira apps you love on Server are not available on Cloud, or maybe you'd like to clean up your data before migration. All these extra decisions, and potentially additional tasks, can add months to your migration timeline.

If you're thinking of moving to Cloud, Atlassian offers a free Cloud migration demo[599] and a free Cloud migration trial[600]. If you've already committed to Cloud, Atlassian urges teams who may require their support to contact them no later than 2 months before your intended migration date.

599 https://www.atlassian.com/migration/plan/cloud-migration-demo
600 https://www.atlassian.com/migration/assess/cloud-migration-trial

9.1.5.2 We're moving to Jira Cloud. How's the Jira Migration Cloud Assistant (JMCA) these days?

Atlassian has continually improved the JMCA experience, so there has never been a better time to migrate to Cloud. Our Server customers now see an 80% success rate when migrating Structure to Cloud using JMCA.

You can find a step-by-step guide in our documentation[601], and on the Atlassian Community[602].

9.1.5.3 If JMCA doesn't work for us, are there other alternative ways to migrate?

While we recommend going the JMCA route, it is also possible to migrate manually to Cloud. You can find the full guide here[603].

9.1.5.4 Are there any manual efforts required, even when using JCMA?

Yes, because of two main reasons:

- Some data is unsupported by Atlassian and can't be migrated by JCMA. You can find the full list of unsupported data here[604].
- You need to migrate everything when using JCMA, not project by project. If you migrate parts without using JCMA, it will be impossible to map the Structure data to Jira data, and this will need to be manually recreated on your cloud instance.

9.1.5.5 What common problems about migration do you hear about?

Each migration is different, but we hear two common problems:

- Jira's user permission configurations are not properly migrated or recreated on their cloud instance, which leads to users not having access to structures or seeing empty structures.
- JQL expressions may not function properly after a migration via JCMA, requiring some additional manual work to recreate some generators and formulas on Cloud.

To help avoid these problems – or catch them earlier than later – we highly recommend using JCMA and performing post-checks on permission schemes and groups.

9.1.5.6 Do you offer a demo instance for customers to test before moving to Cloud?

We do not offer a demo instance for Structure Cloud, but you can try Structure for Jira Cloud free for 30 days and extend the trial by writing to Tempo Support[605].

For Jira Software, Jira Service Management, and Confluence users with Premium and Enterprise plans, Atlassian offers a product sandbox where you can add apps, including Structure. You can read more about the product sandboxes here[606].

---

601 https://wiki.almworks.com/documentation/structure/latest/cloud/migrating-to-cloud-using-the-jira-cloud-migration-assistant-152895884.html

602 https://community.atlassian.com/t5/Marketplace-Apps-Integrations/How-to-Migrate-Structure-App-Data-to-Jira-Cloud-Common-Pitfalls/ba-p/1952950

603 https://wiki.almworks.com/documentation/structure/latest/cloud/manually-migrating-to-cloud-130884950.html

604 https://support.atlassian.com/migration/docs/what-gets-migrated-with-the-jira-cloud-migration-assistant/

605 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12512

606 https://support.atlassian.com/organization-administration/docs/manage-product-sandboxes/

9.1.5.7 When are you going to add Effectors/Advanced Formulas/Transformations to Cloud?

We are continuously striving for feature parity between Server and Cloud. We've released the most business-critical features like Formulas and Quick Filters in 2022, based on customer feedback. Other features like Advanced Formulas, Effectors, and Transformations – along with others – remain on our backlog. We encourage customers to share their feedback and describe their use cases by writing to product@almworks.com[607].

You can find our Structure Cloud roadmap here[608], which includes Server features like Time in Status column, Confluence Macro, and more.

9.1.5.8 Does Structure store any data?

Structure doesn't store any Personal Identifiable Information, but it does store data related to the configuration of the actual structures, such as:

- Structure names, descriptions, and permissions
- Hierarchies (referring to individual issues by their numeric IDs, without any field data; when Group automation is used, numeric IDs of grouped items, such as priority IDs, may also be stored)
- Labels (when Group by Labels automation is used)
- Group names for "Group Picker (single group)" custom fields (when those fields are used in a Group by Field generator)
- Folders
- Column configurations
- Generator configurations (link IDs, issue type IDs, field IDs, JQL and Text queries)

You can also refer to our full list of data stored[609] by Structure Cloud servers.

9.1.5.9 How secure is Structure Cloud?

Structure was one of the first apps[610] to be part of Atlassian's Cloud Fortified[611] program. Cloud Fortified apps adhere to Atlassian's highest, advanced standards for security, reliability, and support.

According to Atlassian, as a Cloud Fortified app, Structure participates in "all six of Atlassian's cloud app security programs and undergoes additional checks for service reliability and performance". Structure integrates incident and review processes with Atlassian to allow for faster recovery time and continuous improvement, and abides by strict Atlassian-defined app support SLAs.

In addition to being accredited as Cloud Fortified, Structure is also a participant in Atlassian's Bug Bounty Program[612] to support our efforts in improving the security of our apps.

Structure also adheres to special SLAs should security-related issues arise. You can find those SLAs here[613].

---

607 mailto:product@almworks.com
608 https://wiki.almworks.com/documentation/structure/latest/cloud/structure-roadmap-152896156.html
609 https://wiki.almworks.com/documentation/structure/latest/cloud/structure-cloud-security-and-privacy-faq-130885108.html
610 https://community.atlassian.com/t5/Marketplace-Apps-Integrations/Introducing-Cloud-Fortified-meet-the-first-Cloud-Fortified-apps/ba-p/1749111
611 https://marketplace.atlassian.com/categories/cloud-fortified-apps
612 https://www.atlassian.com/licensing/marketplace#licensing-purchasing
613 https://wiki.almworks.com/documentation/company/security-bug-fix-policy-120132363.html

9.1.5.10 Can I take Structure.Gantt charts to the Cloud?

Yes, you can! You can find the JMCA migration guide here (pending Structure.Gantt release), or the steps on how to migrate Gantt charts manually here[614].

9.1.5.11 Is Structure GDPR-compliant? What is your GDPR policy?

Yes, Structure for Jira Cloud is GDPR compliant. Structure does not collect, store, or process Personal Identifiable Information.

You can find the Tempo Group's GDPR policy here[615].

9.2 Migrate from Server to Data Center

If you're migrating from Jira Server to Jira Data Center, please follow the instructions below to migrate your Structure configurations and all your structures:

- Using Your Existing Hardware(see page 940)
- Moving to New Hardware(see page 940)

9.2.1 Using Your Existing Hardware

This is the easiest way to migrate to DC. If you've installed a DC instance on the same hardware as your Server instance, the Structure software and all your structures should already be there (Structure and all extensions data is stored in the Jira database). You will just need to update your license (from Server to Data Center) for Structure and any installed extensions (Gantt, Pages, etc.).

Need help or have questions? Contact Tempo Support[616].

9.2.2 Moving to New Hardware

If you're installing Jira Data Center on new hardware or moving Jira data project by project, you'll need to back up your Structure data and import it to the new system. The following guides will walk you through each step:

1. Backing Up Structure(see page 962)
2. Migrating Structures(see page 964)

Migration is available for Structure and Structure.Gantt, and any other extensions (Pages, Deliver, etc.) will be included in the backup.

Need help or have questions? Contact Tempo Support[617].

---

614 https://wiki.almworks.com/documentation/structure/latest/cloud/manually-migrating-to-cloud-130884950.html
615 https://www.tempo.io/tempo-and-gdpr-compliance
616 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514
617 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

9.2.3 FAQ - Migrating to Data Center

9.2.3.1 How long does a migration from Jira Server to Jira Data Center take?

It depends.

Factors like the size of your Server instances, how many Jira third-party apps you need to move, and the human resources available to you – whether it's in-house help from your Jira admins or Atlassian Marketplace Solution Partners – are straightforward, but others may not be so obvious upfront. Perhaps the Jira apps you love on Server are not available on Data Center, or maybe you'd like to clean up your data before migration. All these extra decisions, and potentially additional tasks, can add months to your migration timeline.

9.2.3.2 What efforts are needed to move Structure from Jira Server to Jira Data Center?

Migrating Structure from Jira Server to Jira Data Center is quite a straightforward process that only requires backing up your old Server instance and then restoring it on the new Data Center instance.

You can find the documentation here[618].

9.2.3.3 What about Structure.Gantt, Structure.Pages, Structure.Testy, Structure.Deliver and Colors for Jira?

For Structure.Gantt, the migration is the same as Structure. It can be migrated via Jira database backups, or Structure's backup and restore feature.

For Structure.Pages, Structure.Deliver, Structure.Testy, and Colors for Jira, the only option is to migrate using Jira database backups.

9.2.3.4 Does Structure store any data?

Structure doesn't store any Personal Identifiable Information, but it does store data related to the configuration of the actual structures, such as:

- Structure names, descriptions, and permissions
- Hierarchies (referring to individual issues by their numeric IDs, without any field data; when Group automation is used, numeric IDs of grouped items, such as priority IDs, may also be stored)
- Labels (when Group by Labels automation is used)
- Group names for "Group Picker (single group)" custom fields (when those fields are used in a Group by Field generator)
- Folders
- Column configurations
- Generator configurations (link IDs, issue type IDs, field IDs, JQL and Text queries)

618 https://wiki.almworks.com/documentation/structure/latest/data-center-and-server/structure-backup-restore-and-migration-32222984.html

9.2.3.5 How secure is Jira Data Center?

Structure for Jira Data Center and all its extensions are Data Center-approved apps, which means they adhere to Atlassian's "DC-readiness" program, which ensures that performance tests are regularly conducted to provide a performant, scalable user experience on Jira Data Center.

They follow all of Atlassian's guidelines for Data Center app development[619], and have undergone Atlassian's DC Apps Security Scanner[620], which is "an internal tool that runs code analysis tools against all marketplace Data Center apps and provides feedback to developers to help them improve the security of their applications." By taking part in this program, Structure subjects itself to regular security procedures.

Structure is also a participant in Atlassian's Bug Bounty Program[621], which means the app is routinely checked for vulnerabilities by penetration testers, to support our efforts in improving the security of our apps.

Structure also adheres to special SLAs should security-related issues arise. You can find those SLAs here[622].

9.2.3.6 Is Structure GDPR-compliant? What is your GDPR policy?

Yes, Structure for Jira Data Center is GDPR compliant. Structure does not collect, store, or process Personal Identifiable Information.

You can find the Tempo Group's GDPR policy here[623].

9.3 Monitoring and Troubleshooting Structure Usage

The best way to check statistics on Structure usage, see structure sizes and assess load time is to check the Performance Audit Log (PAL). To open it, go to **Administration | Structure | Support | View Performance Audit Log**.

> ⚠ If you have a Data Center instance, you will need to collect and analyze the Performance Audit Log from every node. When a user opens a structure while being on a certain node, structure will execute all the required operations on that node.

There is no built-in parser, but you can write a simple script to parse the PAL files, search for a structure in the file based on the criteria you need and send alerts if certain problems are found.

9.3.1 Reading the Performance Audit Log

The following breaks down how the PAL file is structured.

9.3.1.1 Basic Instance Information

Lists the version of Jira and Structure.

---

619 https://developer.atlassian.com/platform/marketplace/guidelines-for-data-center-app-development/
620 https://developer.atlassian.com/platform/marketplace/dc-apps-security-scanner/
621 https://www.atlassian.com/licensing/marketplace#licensing-purchasing
622 https://wiki.almworks.com/documentation/company/security-bug-fix-policy-120132363.html
623 https://www.tempo.io/tempo-and-gdpr-compliance

9.3.1.2 Structures Descriptions

This section contains descriptions of structures, which includes:

- Structure names and IDs
- Generators with all their parameters
- Synchronizers with all their parameters
- If Manual Adjustments(see page 194) are enabled for a structure, information on such adjustments
- If any saved filters are used in the generators or synchronizers, the JQLs for such filters

9.3.1.3 Forest Caches

This section details recently accessed structures. It includes the following information:

- structureId - id of the structure
- count of rows - total number of items in the structure, including generators and folders
- unique items count - number of unique items
- item types - number of items by type (how many of the items are issues, generators, folders or loop markers)
- counts on depths - number of items on each level with the first number being the top level

When reviewing the Forest Caches, you should look for:

- The sizes of structures. Larger structures (tens of thousands of rows) can impact performance and are often avoidable. If you're seeing a lot of large structures, or frequent use of large structures, it might make sense to check with the users and see if there is an easier (or more efficient) way to view the same information. See Building Efficient Structures With Automation(see page 942).
- The number of rows vs unique items. If the number of rows is much bigger than the number of unique items, it means there are multiple duplicates in this structure. Sometimes this is necessary, but in many cases it indicates the structure was built incorrectly or inefficiently. See Building Efficient Structures With Automation(see page 942).
- In the types section, you may see "type-loop-marker" records. This indicates that the generators in a structure are producing a loop. This could simply be the result of visualizing a link relationship where there is a cyclical link. Or it could indicate a larger problem with the data or an incorrect configuration, which could result in multiple loop markers. In such cases, it's a good idea to check why they are there. If 5-10% of all items are loop markers, it is recommended that you carefully review the structure configuration.

9.3.1.4 Forest Changes Updates

This section shows the time in ms it took to load a structure or update the one already open. Full update means the structure was fully reloaded. Incremental update means some changes happened in Jira, and a part of the structure was adjusted.

We recommend carefully monitoring loading times. If these times become very large, it's an indication that these structures may require attention. Larger structures will naturally require more loading time (sometimes over a minute for very big and complex structures), but you should still monitor for any changes - if loading times suddenly became much longer, this may indicate a problem.

9.3.1.5 Attribute Service

This section contains some technical information used by ALM Works for further troubleshooting.

9.3.1.6 Saved Filters with S-JQL

If you are using S-JQL(see page 428) in any saved filters, such filters will be listed here.

Whenever an S-JQL query is executed, the structure it references must be generated. This means every time the saved filter is used (either explicitly or, for example, on the Agile board), the associated structure is generated, whether users are actually opening the structure or not. This may affect performance, particular if the S-JQL is referencing large structures.

9.3.1.7 Gantt Settings

This section lists those structures that have Gantt charts configured and the configuration details.

9.4 Installing Structure

Structure is installed like most other apps.

1. Before installing Structure in production, make sure your Jira meets the Memory Guidelines(see page 944).
2. Open Manage add-ons, search for "Structure" by ALM Works on the Atlassian Marketplace and install from there.

   Alternatively, you can download the plugin JAR manually from the download page(see page 1164) and either place it into the *plugins/installed-plugins* subdirectory under your Jira home (then restart Jira) or use the "Upload Add-on" link in Manage add-ons.

3. Press the **Get Started** button to finish the installation by installing a license key(see page 950).

Congratulations! You can now spread the word and help users get started with Structure – see Getting Started with Structure(see page 944).

If Structure Remains Disabled

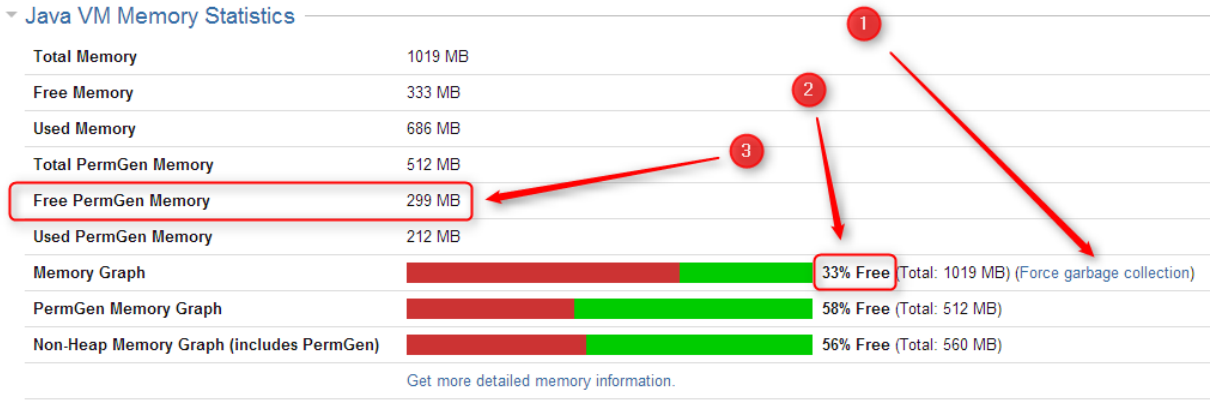
It is possible that after you install Structure or enable it from Manage add-ons, the add-on will remain disabled. An error may or may not be shown. If you refresh the Manage add-ons page within 5-10 seconds and Structure is still disabled, see Structure Won't Start(see page 673) for possible causes and solutions.

9.4.1 Memory Guidelines

On a production system, it is a good idea to check if you have enough free memory in Jira's Java process before installing Structure (or any other plugin).

9.4.1.1 Assessing Available Memory

1. Open menu **Administration | System | System Info** and scroll down to **Java VM Memory Statistics**.
2. Click **Force Garbage Collection**
3. Note the free % number of the **Memory Graph** (heap memory).
4. Note the absolute amount of **Free PermGen Memory** (non-heap memory for Java classes).



Memory Statistic	Recommended Value	Parameter in `setenv.sh` / `setenv.bat`
% of Free Heap Memory	25% – 50%	`JVM_MAXIMUM_MEMORY`
Free PermGen Memory (prior to Java 8)	100 – 200 MB	`JIRA_MAX_PERM_SIZE`

If you run Jira on Java 8, PermGen memory is not a factor.

All recommendations are for a general case and do not guarantee that you won't get `OutOfMemoryError`. Individual cases may vary.

9.4.1.2 Heap Memory Requirements

It is recommended that % of free heap memory is from 25% to 50%.

Structure requires about an additional 100 MB of heap memory. You can take your current statistic of **Used Memory** and **Total Memory**, add 100 MB to the **Used Memory** and calculate the recommended value for the **Total Memory**.

If you already have the recommended % of free memory, you can just increase total heap memory by 200 MB.

9.4.1.3 PermGen Memory Requirements

This section applies to Jira running on Sun/Oracle Hotspot Java VM only.

PermGen space is used for Java classes and may be depleted if you uninstall, install or upgrade plugins frequently, or if you don't restart Jira over a long period of time. Due to technical reasons, PermGen space might not get cleaned up from the obsolete classes, and you may end up with an `OutOfMemoryError: PermGen space` error.

Structure classes use only about 10 MB of PermGen space. But for the reasons just mentioned, it is good to have a safety margin with a free PermGen space of at least 100 MB.

9.4.1.4 Changing Memory Parameters

To change memory parameters, edit `setenv.sh` (on Windows, `setenv.bat`).

- To change the maximum amount of Heap space, edit the `JVM_MAXIMUM_MEMORY` parameter near the top of the script.

```
JVM_MAXIMUM_MEMORY="2000m"
```

- To change the maximum amount of PermGen space, edit the `JIRA_MAX_PERM_SIZE=256m` line. Alternatively, you can add the `MaxPermSize` parameter to `JVM_SUPPORT_RECOMMENDED_ARGS.` For example:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-XX:MaxPermSize=400m"
```

You need to restart Jira for these settings to take effect.

9.4.1.5 Use 64-Bit Java

It is imperative to use 64-bit Java when allocating a large amount of memory to it (1 GB and more). To check if you're running 64-bit Java, look up the **Java VM** parameter on the System Info page.

9.4.1.6 Physical Memory Requirements

Avoid swapping at all costs!

The amount of physical memory should be enough to accommodate the whole heap and non-heap memory. If you have other Java or memory-intensive applications running on the same host, they all should fit in physical memory, plus you need to reserve at least 1 GB for the operating system, services, and file cache.

Do not allocate more memory to Jira if it cannot fit into physical memory! If Java running Jira starts swapping actively used memory, it will severely impact performance.

Sample calculations for a host running Jira and Confluence, with Apache and MySQL:

Jira	Heap: 2 GB Non-heap: 500 MB
Confluence	Heap: 2 GB Non-heap: 500 MB

Operating system Apache HTTPD MySQL	1 GB
Free memory margin / File buffers	2 GB
Total Physical Memory Required	**8 GB**

9.4.2 Uninstalling and Reinstalling Structure

9.4.2.1 Uninstalling Structure

You can uninstall Structure from the Add-on Manager the same way you uninstall other add-ons. You can also manually remove the Structure JAR from the `plugins/installed-plugins` directory when Jira is not running.

When you uninstall the Structure add-on, Structure data is **not removed**. It remains in the Jira database.

9.4.2.2 Reinstalling Structure

It is perfectly safe to uninstall Structure, then install it again. (This happens, for example, when you upgrade to a newer version.)

All Structure data will be there unless you manually remove it.

9.4.3 Upgrading and Downgrading

9.4.3.1 Upgrading

To upgrade Structure 3.0 or later:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure**. See Backing Up Structure(see page 962) for details. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Check Structure extensions. If you are using Structure.Testy, Structure.Pages, Structure.Gantt or other extensions, they may become disabled. In this case, you need to either upgrade them too (it might be a compatibility requirement) or enable them manually in the Add-on Manager. If they fail to enable, reinstall them (uninstall and install again).
5. Check plugins that integrate with Structure. As with extensions, make sure they are enabled and upgrade/ reinstall as necessary.
6. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

For more specific instructions, please check the Release Notes(see page 681) for the version to which you wish to upgrade.

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 909).

9.4.3.2 Downgrading

Reverting Structure to an older version is not always possible, because newer versions can modify the database, making it incompatible with older versions.

Simplified Downgrade

A simple downgrade is possible if the database schema hasn't changed. Check the Release Notes(see page 681) for the version you are downgrading from and look for a downgrade advisory. Proceed only if you have indications that it is safe to downgrade to the specific version you have in mind.

1. Uninstall Structure. This step is required, because Add-on Manager will not install an earlier version over a later version.
2. Install the version that you need.
3. Check Structure extensions and integrating add-ons. See the steps in the Upgrading section above.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors. **This is especially important with this kind of downgrade, because some errors may be subtle and not visible to users!**

Reliable Downgrade

Reliable downgrade requires a Structure backup file and manual access to the database.

1. Create a Structure backup using **Administration | Structure | Backup Structure**.
    a. Backup files are backward / forward compatible along Structure 5.x series. To downgrade to an earlier version, see Downgrading to Structure 3.0 - 4.6(see page 949). Note: it is not possible to downgrade to Structure 2.x using a 3.0 or later backup file.
    b. You can also use a previously created backup file. **Note that all data will be rolled back to the state when the backup file was created.**
2. Uninstall Structure.
3. **Double-check you have the backup! You are about to delete all Structure data.**
4. Manually access your database using database tools. Drop all tables that start with `AO_8BAD1B_`. If after that you have other objects starting with that prefix, drop them too.
5. Install the previous version of Structure.
6. Use **Administration | Structure | Restore Structure** to populate the data from the backup file.
7. Check Structure extensions and integrating add-ons. See the steps in the Upgrading section above.
8. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Creating a backup and restoring from backup may require considerable time. If you want to speed up the process and you don't need the history of structure changes, turn off the option "Include History" when creating a backup.

Downgrading to Structure 3.0 - 4.6

If you are using Structure 3.x or 4.x: follow the upgrade instructions above.

If you are using Structure 5.0 or later: in Structure 5.0 we had to change the backup file format version, due to the backwards-incompatible changes required by the introduction of manual adjustments. This means that previous versions of Structure will not be able to restore data from backup files created by Structure 5.0 and later.

To downgrade to an earlier version, use the procedure outlined below to restore Structure data from a 5.0 backup file:

- Unpack the XML backup file from the ZIP archive created by Structure.
- Change the `version` attribute in the `<structure-backup>` element from "5.0" to "3.3".
- Delete all `<manualAdjustments>` elements from the XML.

Then you can restore directly from the modified XML file; you do not have to pack it into a ZIP archive.

9.4.4 Migrating Data from Structure 2 to Structure 3

Unlike previous versions, Structure 3.0 uses the main Jira database to store its data. You need to migrate the data from Structure 2 in order to continue working with it in Structure 3. Additionally, this feature can be used to restore structures from a backup made with Structure 2.

Structure 2 had a separate Backup / Restore functionality, because Structure data was kept separately. With Structure 3, all data is backed up with the usual Jira backup.

However, we plan to reinstate Backup / Restore / Migrate feature in future versions of Structure 3.

9.4.4.1 Creating a Backup of Structure 2.x Data

- If you still have Structure 2.x installed, create a backup of the current Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the entire `structure/` sub-directory under Jira home while Structure is disabled. See Backing Up Structure(see page 962) for details.
- If you already have Structure 3.x installed, use the **Administration | Structure | Export Structure 2.x Data** page. It allows you to create a backup zip with all Structure 2.x data and then opens the **Restore Structure** page, allowing you to immediately import the backup into the Structure 3.x database.

9.4.4.2 Restoring Structure Data from 2.x Backup

1. Use the **Administration | Structure | Restore Structure** menu and use any Structure 2.x backup made earlier. Note that it should be placed in the `import/` directory on your server.
2. If you used the "Export Structure 2.x Data" menu, you will be taken to the restore automatically.

9.4.4.3 After Data Migration

Upgrade Testy

If you have Structure.Testy installed, download and install the latest version of Structure.Testy, compatible with Structure.

Upgrading "Global Structure"

If you're using a "Global Structure" structure, which was created by default in Structure 2.x, you need to make sure that there's an "owner" of that structure. Otherwise, Generators(see page 140) will not work there.

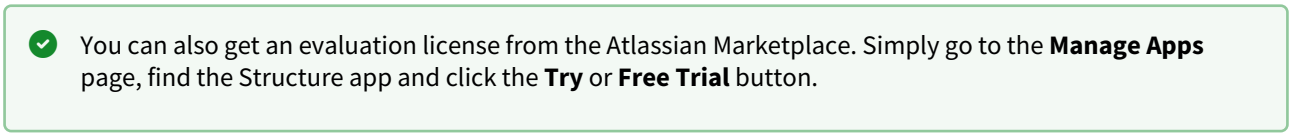
1. Open **Structure | Manage Structure**.
2. Find Global Structure and check if it has an Owner assigned.
3. If it doesn't have an owner, click **Configure** and set yourself as the owner.

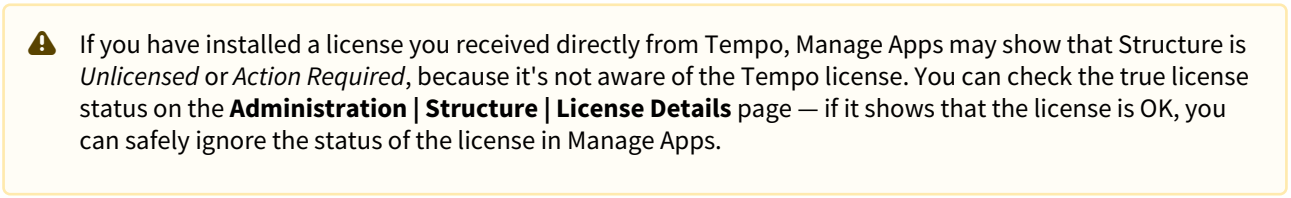
9.5 Setting Up Structure License

Unless your Jira runs on one of the free licenses(see page 953), Structure requires a license key to work. You can get a free, no-obligation 30-day evaluation license key for your Jira server in a few seconds.

9.5.1 Setting Up Evaluation License

1. Navigate to **Administration | Structure | License Details**.
2. Look at the **Current License** section - if there's no license there or if the license is expired, you will need to get an evaluation license or purchase a commercial license.
    - If the **Current License** section says you have a **Free License**, then your Jira must be qualifying for automatic free license and no further action is needed from you. See When Structure is Available for Free(see page 953).
3. To get a free 30-day unlimited-users evaluation license, follow the **Get Evaluation License** link on the Structure license page, or open the evaluation license request page[624] directly. In the latter case, please enter your Jira Server ID to ensure you receive a correct license.





9.5.2 Licenses from Tempo and from Atlassian

Structure supports two kinds of licenses — issued by Tempo and issued by Atlassian. These licenses are functionally equal — you can use either kind to get the same functionality in Structure. The prices are also the same.

The following table summarizes the differences and provides instructions for both kinds.

624 http://almworks.com/structure/evaluate.html

	License from Tempo	**License from Atlassian**
Purchased at	No longer available	Atlassian Marketplace[625]
Managed at	The license key is sent to you by email	Manage with other Atlassian licenses at my.atlassian.com[626]

License key looks like this:	`-----BEGIN CERTIFICATE-----`
	
	`MIIEYTCCAkmgAwIBAgIGAT2oPFqOMA0GCSqGSIb3DQE...`
	` ... at least 20 lines of symbols ...`
	` -----END CERTIFICATE-----`
		`AAABEA0ODAoPeNp9UE1Pg0AUvO+v2MSbCc0uQZO...`
		` ... at least 4 lines of symbols ...`

Installation Instructions	1. If you have a license from Atlassian installed, first remove it in **Manage Apps**. 2. Open **Administration** | **Structure** | **License Details**. 3. Copy and paste the key to the **Install License** section and click **Install License**.	1. Open **Manage Apps**. 2. Locate and open the Structure section. 3. Copy the license key into the **License** box and click **Update**.
Uninstallation Instructions	1. Open **Administration** | **Structure** | **License Details**. 2. See the details of the installed license and click **Uninstall**.	1. Open **Manage Apps**. 2. Locate and open the Structure section. 3. Clear the license key from the **License** box and click **Update**.
Purchasing differences	• Besides advance payments with credit card, wire transfer or other payment methods supported by our payment processor, we can also accept purchase orders on Net 30 terms. • VAT and taxes may be handled differently from Atlassian, as our payment processors are located in the USA and Germany. Tempo is based in the US, and for direct purchases using Wire Transfer, we do not charge VAT or any other taxes.	• Purchasing from Atlassian is not available in certain countries[627].

625 http://almworks.com/structure/marketplace.html
626 https://my.atlassian.com
627 http://www.atlassian.com/licensing/marketplace#generalmarketplacequestions-11

9.5.3 Purchasing a Commercial License

Structure licenses can be purchased from Tempo, from Atlassian, or through Atlassian Solution Partners and resellers.

9.5.3.1 Purchasing from Atlassian

You can purchase a license via Atlassian on the Atlassian Marketplace[628].

After the purchase is completed, the license key will be available on https://my.atlassian.com.

9.5.3.2 Purchasing from Resellers or Atlassian Experts

You can purchase through a reseller of your choice. Atlassian Solution Partners[629] can also provide you with additional services and advice.

When you purchase through a reseller, you can get either kind of license (issued by Tempo or by Atlassian), depending on the reseller's actions. If you prefer one kind of license over another, you should specify that to the reseller.

9.5.4 Migrating Licenses

You can convert a license of one kind into a license of another kind. Please contact sales@almworks.com[630] for assistance.

Next: Select which projects are enabled for Structure(see page 954)

9.5.5 Structure License Parameters

The following parameters are displayed in the **Current License** section when you install a Structure license.

Parameter	Meaning
License Type	Commercial, Evaluation or other
Licensee	Organization authorized to use the license
Serial Number	A unique number assigned to the license

628 http://almworks.com/structure/marketplace.html
629 https://www.atlassian.com/partners
630 mailto:sales@almworks.com

Parameter	Meaning
Expires	If present, the license is not perpetual: it will expire at the specified date. After that date passes, Structure will not be available unless the license key is changed.
Maintenance Expires	If present, the license key can only work with the versions of Structure released prior to the specified date. If you need to use a newer version of Structure, you will need to renew maintenance.
User Limit	This is the maximum number of users allowed by Jira that are supported by this license key. The license that Jira runs on must allow this number of users or fewer.
Server ID	Although not shown in the license table, most licenses are tied to a specific Jira server ID and will not install on a server with a different ID. If you need to move a license key to a different server, please contact support.

9.5.6 When Structure is Available for Free

Structure automatically installs a free license if your Jira instance runs on one of the following free licenses:

- Free license for **open-source** projects
- Free license for a **non-profit** organization
- Free **community** license
- Free **demonstration** license
- Free **developer** license

The clauses from the Atlassian EULA that govern the use of those free licenses also apply to using Structure on Jira servers where these licenses are installed.

9.5.7 License Maintenance and Expiration

9.5.7.1 Commercial License

Your commercial license for Structure (including Starter licenses) typically has no expiration date, so it can be used forever. However, it has *Maintenance Expiration Date* which limits which versions of Structure can be used with that license – you can only use the versions released prior to the expiration date.

To use versions released later, you need to purchase a maintenance renewal, which extends your maintenance expiration date by one year.

9.5.7.2 Evaluation License

Evaluation and temporary licenses have an expiration date. After that date, Structure will only be accessible in read-only mode (see below).

Make sure you renew the evaluation or get another license key prior to the expiration.

9.5.7.3 Read-only Mode

If the current license becomes invalid (for example, it has expired or you upgraded to a version of Structure that is not covered by the current license's maintenance), Structure will function in read-only mode.

Users will be able to view structures, but they won't be able to make any changes until a valid license is installed.

9.6 Selecting Structure-Enabled Projects

Structure can be enabled for any selection of Jira projects, or for none of them.

> ⚠ By default, Structure is enabled for all projects. To limit users' exposure to Structure, pick specific projects to be enabled for Structure.

To select which projects are enabled for Structure:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Enable/Disable Structure in Projects**.
3. Select whether Structure should be available for **all projects** or for **selected projects**.
4. In the latter case, update the **Selected Projects** list by selecting one or more projects and using the **Enable** and **Disable** buttons.
5. Click **Apply** when done.
6. If you disable projects that are already used in a structure (a structure contains issues from that project), you will be given a warning. You can opt to **Proceed with Changes** or cancel.
    a. If you proceed and disable a project that has issues in some structures, those structures will appear to the users without those issues.
    b. If you later enable that project again, the issues will reappear where they were (all structure changes taken into account).

> ✓ Which projects are enabled for Structure affects Who Has Access to the Structure(see page 955)

9.7 Global Permissions

9.7.1 Who Has Access to the Structure

Structure is visible only to specific users. Only users who have access to Structure will see the *Structure* menu and other user interface elements provided by Structure.

A user has access to Structure if all of the following conditions are met:

- The user has **Browse** permission on at least one of the projects that are enabled for Structure(see page 954).
- Structure is enabled for this user(see page 955):
  - Either Structure is enabled for everyone, or
  - The user belongs to at least one of the enabled groups, or
  - The user belongs to at least one of the enabled roles in an enabled project.

Users who have *Jira Administrators* global permission always have access to Structure.

9.7.2 Restricting User Access to Structure

By default, Structure is accessible to anyone who has *Browse* permission on structure-enabled projects(see page 954). You can further restrict this access level to one or more user groups.

To select who can use Structure:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Select Structure Users**.
3. Select whether Structure should be available to **Everyone** or to **Users in selected groups/roles**.
4. In the latter case, change the **selected groups/roles** list by selecting the second radio button and use the **Add Group/Role** section to add one or more required user groups or project roles. To set up required property, use the drop-down selectors to choose either **Group** or **Project** option, then choose the required group name or project/role combination and press the **Add** button to add it to the list. If **project** is set to "Any", this means that the user should be in the specified role for any of structure-enabled projects(see page 954).
5. You can remove the permission option by clicking the trashcan icon on the right of the option.
6. Click **Apply** when done or **Cancel** to dismiss your changes.

Which projects are enabled for the Structure also affects Who Has Access to the Structure(see page 955).

When Structure is enabled for **anyone**, even anonymous visitors will have access to Structure. To make Structure accessible to only logged in users, restrict access to the **jira-users** group.

9.7.3 Changing Permission to Create New Structures

By default, any logged-in user with access to Structure(see page 955) can create new structures of their own. However, you can restrict this ability to one or more user groups.

To select who can create new structures:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Select Who Can Create Structures**.

3. Select whether new structures can be created by **Anyone with access to Structure** or by **Users in selected groups/roles**.
4. If permissions are based on groups/roles, use drop-down menu to choose either **Group** or **Project**, and then select the required group name or project/role combination. To search the list, simply select the field and begin typing. Click **Add** to include the selected group/role. *Note:* If **project** is set to "Any", this means that users should be in a specified role for any of structure-enabled projects(see page 954).
5. You can remove permission option by clicking the trash can icon on the right of the option.
6. Click **Apply** when done or **Cancel** to dismiss your changes.

The user also needs general access to Structure(see page 955) to be able to create new structures.

Users who have *JIRA Administrators* global permission are always allowed to create new structures.

9.7.4 Changing Permission to Access Automation

By default, any user with Edit Generators access level(see page 543) for a structure can add and configure generators(see page 140). You can restrict this ability based on user groups and/or project roles.

To select who can edit generators:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Select Who Can Access Automation**.
3. Select whether generators can be changed by **Anyone with Edit Generators permission** or by **Users in selected groups/roles**.
4. In the latter case, change the **selected groups/roles** list by selecting the second radio button and use the **Add Group/Role** section to add one or more required user groups or project roles:
    a. Use the drop-down selectors to choose either the **Group** or **Project** option.
    b. Choose the required group name or project/role combination. (*Note:* If **project** is set to "Any", this means the user should be in the specified role for any of the structure-enabled projects(see page 954).)
    c. Press the **Add** button to add it to the list.
5. You can remove a permission option by clicking the trash can icon on the right of the option.
6. Click **Apply** when done or **Cancel** to dismiss your changes.

In order to add/change generators for an individual structure, a user also has to have *Edit Generators* access (see page 543)for that structure.

Users who have *Jira Administrators* global permission are always allowed to change generators.

9.7.5 Changing Permissions to Configure and Run Effectors

There are three levels of permissions for Effectors(see page 179):

- **Permission to Configure Effectors** - by default, anyone with access to Automation(see page 956) can add and configure Effectors.

- **Permission to Execute Effectors** - by default, anyone with view access to a structure (except anonymous users) can run any Effectors within the structure.
- **Permission to Execute Effectors on Query Results** - this permission is required to run effectors that were inserted by a Structure Inserter(see page 146).

In order to run an Effector, user must have Bulk Change global permission in Jira.

You can restrict users' ability to access/run Effectors based on user groups and/or project roles. To do so:

1. Navigate to **Administration | Structure | Configuration**.
2. Scroll to the appropriate Effector permission at the bottom of the page. Click the appropriate "Select who can..." button.
3. If you choose to assign permission based on **Users in selected groups/roles**, use the **Add Group/ Role** section to add one or more required user groups or project roles:
    a. Use the drop-down selectors to choose either the **Group** or **Project** option.
    b. Choose the required group name or project/role combination. (*Note:* If **project** is set to "Any", this means the user should be in the specified role for any of the structure-enabled projects(see page 954).)
    c. Press the **Add** button to add it to the list.
4. You can remove a permission option by clicking the trash can icon on the right of the option.
5. Click **Apply** when done or **Cancel** to dismiss your changes.

Users who have *Jira Administrators* global permission are always allowed to create, edit and execute Effectors.

9.8 Changing Structure Defaults

Jira administrators can adjust a number of Structure "defaults", settings that apply when the user does not specify a more specific request or option.

- Changing Default Structure(see page 958)
- Changing Default View Settings(see page 959)
- Changing Default Options for the Issue and Project Pages(see page 959)

9.8.1 Initial Configuration

When Structure is installed, the defaults are configured as follows:

System Default Structure	None
Project Default Structure	None
Default Views Menu	Preinstalled views: **Basic**, **Planning**, **Tracking**, **Triage**, **Entry** (on all pages)

Default View	Basic View
Auto-switch (Issue Page)	Structure with the displayed issue.
Auto-switch (Project Page)	**Off** - Show the last viewed structure.
Keep structure when navigating	**On** - When going from structure widget to an issue page, show the same structure.
Auto-minimize Structure Panel	**On** - Structure panel is initially minimized if the issue is not in the structure.

9.8.2 Changing Default Structure

The default structure(see page 537) is automatically selected when the user opens Structure Board(see page 88) for the first time. It is also shown when a user views the Structure widget from an issues page, if the Auto-switch option(see page 93) is set to *default structure*.

You can change the default structure for the entire Jira instance, as well as for specific projects.

9.8.2.1 Changing System-level Default Structure

The system-level default structure is used when no other structure has been selected or the Auto-switch option is set for issue pages (and a project-level default structure has not been specified).

1. Open the **Administration | Structure | Defaults** menu.
2. In the **System Default Structure** section, click **Change**.
3. Select the default structure and click **Apply**.

⚠ Make sure users have View permissions(see page 543) to the default structure. Users without view permissions will receive an error when the default structure should be displayed.

9.8.2.2 Changing Project-level Default Structure

You can select a default structure for specific projects.

1. Open the **Administration | Structure | Defaults** menu.
2. Locate the project in the **Project Default Structures** section and click **Change**. If you can't find the structure, un-check the **Show only projects with overridden default structure** checkbox.
3. Select a structure and click **Change**.
4. Or, select **Use system default** to remove the project-level default.

Project administrator can also change the project-level structure from the **Structure** tab on the project administration page, or from the options pop-up window on the **Structure** tab on the user's project page.

9.8.3 Changing Default View Settings

View settings determine which views are offered to the users in the Views Menu (on the Structure Board and other pages with Structure widget). The default view settings apply to all structures that don't have customized view settings(see page 541).

To change default view settings:

1. Open the **Administration | Structure | Defaults** menu.
2. In the **Default View Settings** section, click **Change**.
3. Modify the default settings - for details, see Customizing View Settings(see page 541).
4. Click Apply

9.8.4 Changing Default Options for the Issue and Project Pages

A number of options define how Structure behaves on issue pages(see page 93) and project pages(see page 95). When the user opens those pages for the first time, the default settings apply. These settings are adjustable by the Jira administrator.

To change the defaults:

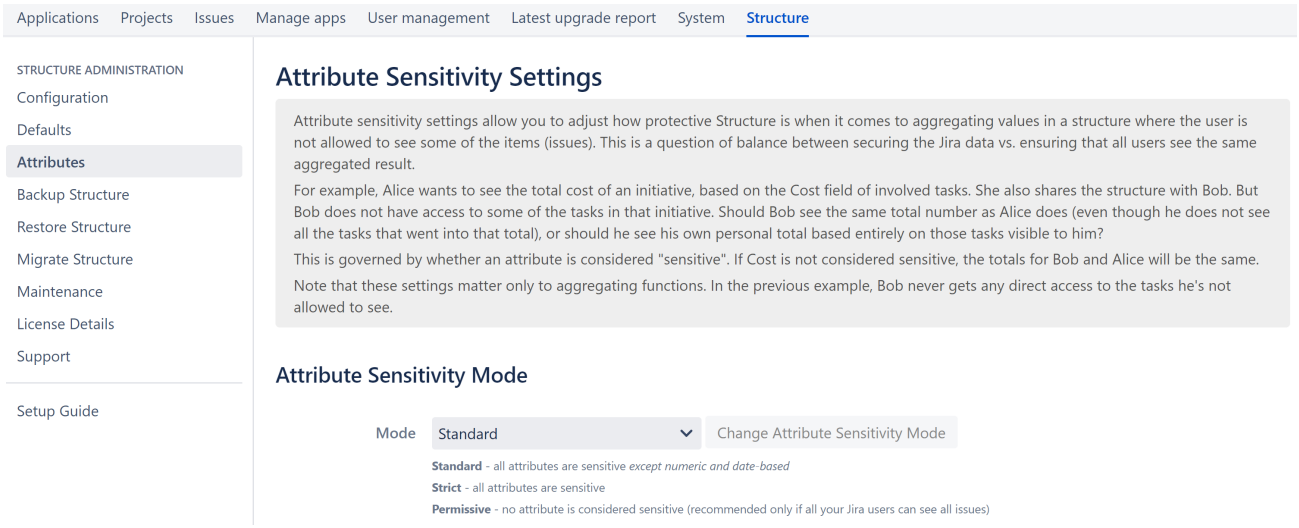
1. Open the **Administration | Structure | Defaults** menu.
2. Scroll down to **Structure User Interface Defaults** and click **Change**.
3. Make the changes and click **Change** again.

Option	Description	See Also
Auto-switch (Issue Page)	Lets you select which structure is initially displayed on issue pages.	Structure Options for the Issue Page(see page 93)
Auto-switch (Project Page)	Lets you select which structure is initially displayed on Project pages.	Structure on the Project Page(see page 95)
Keep Structure Selection When Navigating	If turned on, clicking an issue in the Structure Widget(see page 88) will open that issue's page with the same structure. If this option is turned off, the structure displayed will be based on the Auto-switch setting.	Structure Options for the Issue Page(see page 93)
Auto-minimize Structure Panel	If turned on, the Structure panel on the issue page will be initially minimized if the selected structure does not contain the displayed issue.	Structure Options for the Issue Page(see page 93)

If the user changes these options, those changes are preserved and are applied instead of the defaults for that specific user.

9.9 Attribute Sensitivity Settings

The Attribute Sensitivity Settings page allows you to specify which values may contain sensitive information and, therefore, should or should not be included when calculating aggregated functions.



9.9.1 How Does Attribute Sensitivity Work?

While team members can only see issues they have permissions to view, Structure's aggregate value calculations can include values from issues they do not have direct access to. In many cases, this may be preferred, particularly for numeric values - Imagine if one of your team members is attempting to calculate total Story Points for a project, but only has access to the Story Points data from their own issues. It won't be possible to accurately calculate the story points for the project.

On the other hand, there may be times you don't want users to have access to sensitive data, even as part of an aggregate calculation. Certain financial values, for example, or fields containing sensitive text-based values.

Attribute Sensitivity Settings allow you to set the level of attribute-based data security Structure will use and create exceptions for specific non-sensitive fields.

9.9.2 Setting the Attribute Sensitivity Mode

To set the Attribute Sensitivity mode for Structure, go to **Administration > Structure > Attributes**.

In the Mode field, select one of the following:

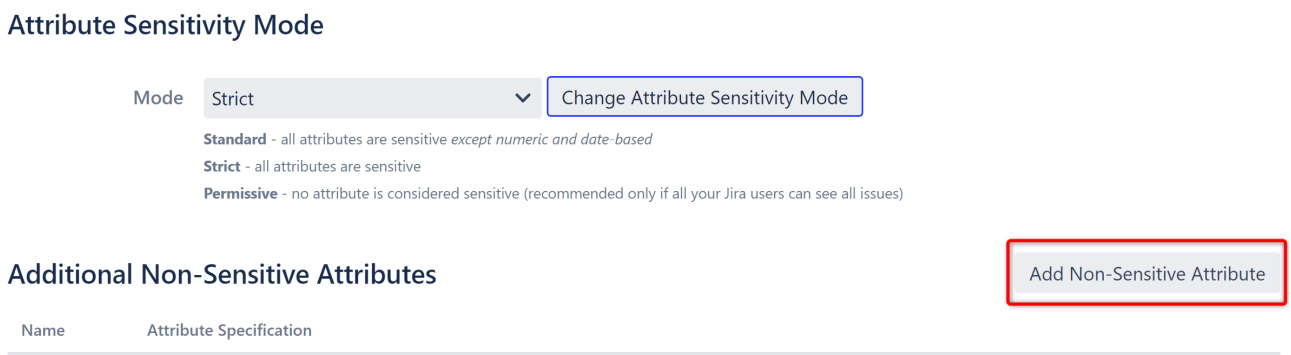
- Standard - sets all attribute fields **except numeric and date fields** to sensitive
- Strict - sets **all** attribute fields to sensitive
- Permissive - sets no attribute fields to sensitive

If an attribute is marked *Sensitive*, aggregate functions that include that attribute will only include values from issues the user has permission to view.

⚠ Permissive sensitivity does not apply to the Last Comment field. When aggregated in a formula, users will still only see comments for issues they have permission to view.

9.9.3 Assigning Specific Non-Sensitive Attributes

You can specify certain attributes to be non-sensitive, regardless of your sensitivity mode. To do so, go to the the Additional Non-Sensitive Attributes section and click the **Add Non-Sensitive Attribute** button.



Next select the attribute you wish to make non-sensitive. When you select from the **Attribute** dropdown, the **Format** and **Spec** details will be updated automatically. You can also enter these manually to customize the attribute requirements or enter an attribute not available in the dropdown.

⚠ **Be careful!** When you add non-sensitive attributes, you allow those values to be included in aggregate results for all issues, even if the user does not have access to an issue. The user will not **see** the specific value for issues they do not have access to, but it may be possible for them to deduce the value by comparing the aggregate value to values they can see.

9.10 Structure Backup, Restore and Migration

Structure data can be backed up and restored separately from other Jira data. Structure data includes structures, hierarchies (forests), generators, synchronizers, folders - everything added to Jira by the Structure add-on. Structure backup does not include issue or other items data (except for some attributes that are added to enable migration.)

You need *Jira System Administrators* global permission to back up, restore or migrate Structure data.

> Starting with Structure 3, when you fully back up Jira, Structure data is also backed up – it is stored in the same database as the Jira data. However, you can use the separate backup to:
>
> - Restore only Structure data, without changing Jira data
> - Migrate structures to other servers (following a Project Import in Jira, for example)
> - Export Structure data to some other tool by parsing the backup XML

9.10.1 Using Structure Backup

Structure can use a backup file in two ways:

- **Full structure restore.** This operation replaces all existing structure data (if any) with the data stored in the backup file. This operation refers to issues and other items by their numeric IDs (*not* issue keys!), so the issues must be present in Jira before this operation is run, and issue IDs must be the same as they were at the time the Structure backup file was created.

  > Issue IDs are preserved if the Jira instance is fully restored from a backup with the **Restore System** command. Issue IDs are *not* preserved if the issues are moved to another Jira instance with Jira's **Project Import** feature – in this case, you should use the migration option below.

- **Migration / partial import.** This operation lets you restore one or more structures backed up from a different Jira instance (assuming the issues have been moved over with the Jira's **Project Import** command). It also allows you to merge the backup structure data with the structure data already existing on your Jira.

  > A structure in a backup file cannot be restored if it refers to issues in a project that is not present in the Jira instance.

- Backing Up Structure(see page 962)
- Restoring Structure from Backup(see page 963)
- Migrating Structures(see page 964)

9.10.2 Backing Up Structure

Backing up Structure saves the existing structures, their configuration, hierarchies and other Structure data. Structure backup does not save the issues themselves or other Jira data - see Structure Backup, Restore and Migration(see page 962).

To back up Structure:

1. Navigate to **Administration | Structure | Backup Structure**.
2. Enter the name for the backup file. If you omit the file extension, *.zip* will be added to it.

   > ⚠ You cannot specify a directory for the backup file. Backup is always done to the *export* sub-directory under Jira home.

3. Use the **Backup History** checkbox to include the full change history in the backup file.
4. Click **Backup**.
5. If the file already exists, you will be given an option to overwrite the file or cancel the operation.
6. You will see the **Process Status** page where you can track the progress of your backup. Once it's finished, click **Show Results** to see the full name of the backup file.

9.10.3 Restoring Structure from Backup

Restoring Structure from a backup brings back the structures, automations, synchronizers, views and other data created at the moment of backup.

> ⚠ Restoring Structure will not affect issues in any way or restore them. The issues that make up the hierarchy should already exist in Jira. If you do full restore, then you need to run the standard Jira data restore first - see Structure Backup, Restore and Migration(see page 962).

> ⚠ The issues and other items in the structures are identified by their internal numeric ID. If you have transferred issues via Jira's Project Import, issue IDs have changed, and you will need to use Structure Migration(see page 964) instead.

Use Restore Structure when all of the following are true:

- The backup was made on this Jira instance or on its predecessor
- You need to fully restore structure data
- You can lose the current Structure data stored on this Jira instance (issues are not affected, only their organization into structures)

To restore Structure from a backup:

1. Navigate to **Administration | Structure | Restore Structure**.
2. Enter the full path to the Structure backup file (either *.xml* or *.zip*).
3. Click **Restore**.
4. If Structure currently has any data, it will ask you to confirm the restore operation. Restoring from backup clears all Structure data, and it cannot be undone! If you have data that you're overwriting, you might want to back up that data first.
5. You will see the Process Status page, which will show you the progress of the restore operation. You can abort the process by clicking the **Abort** button on the status page.

   > ⚠ If you abort the restore operation, Structure data will be left in a partially restored state. You may see some of your structures, but not all of them, and auxiliary data like synchronizers, views, favorites and perspectives may be completely lost. You can revert to the original state only by fully restoring Structure from another backup.

6. Once the process is finished, the **Show Result** button will take you to the result page, where you'll be able to see the result and any warning messages.

After Structure has been restored, open the **Structure | Manage Structure** page to verify that your structures were restored successfully.

9.10.3.1 Downgrading from Structure 5.0 or Later

The introduction of Manual Adjustments(see page 194) in Structure 5.0 required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later.

To downgrade to an earlier version, you first need to restore Structure data from a 5.0 backup file:

1. Unpack the XML backup file from the ZIP archive created by Structure.
2. Change the `version` attribute in the `<structure-backup>` element from "5.0" to "3.3".
3. Delete all `<manualAdjustments>` elements from the XML.

Once completed, you can restore directly from the modified XML file using the procedure above. You do not have to pack it into a ZIP archive.

9.10.4 Migrating Structures

Migrating structure data lets you import one or more structures from a different Jira instances after you have imported projects with Jira's Project Import operation. In addition, you can add structures from a backup file to those that are already present in Jira.

⚠ Migrating structure will not affect issues in any way. The issues that make up the hierarchy should already exist in Jira. You may need to run Jira Project Import or the standard Jira data restore first - see Structure Backup, Restore and Migration(see page 962).

⚠ During migration, the issues in the structures are located in Jira by their issue keys. A structure cannot be migrated if it refers to issues from a project that is missing in Jira.

To migrate structures from a backup:

1. Navigate to **Administration | Structure | Migrate Structure**.
2. Enter the full path to the structure backup file (either *.xml* or *.zip*).
3. Click **Select Structures To Restore**.
4. Select the structures that should be restored. If there's an existing structure with same ID or name, select **Overwrite Existing** to replace the existing structure with the one from backup; otherwise the structure will be restored as a new structure, leaving the existing one unaltered.
5. Under the list of structures, you will see a list of additional restore options:

Restore Structure Permissions	If selected, Structure will attempt to restore the access permissions for the imported structures. This attempt may fail if the permission rules refer to users or groups not present in Jira. If you don't select this option, or if the attempt to restore permissions fails, the restored structure will have no permission rules. Jira administrators can configure permission rules through the Manage Structures(see page 532) page.

Restore Synchronizers	If selected, the synchronizers for selected structures are restored. ⚠ Synchronizers configuration is imported as-is, and might not make sense on a new Jira instance. After you have restored synchronizers, please visit the Synchronization Settings(see page 983) page to check if the synchronizers are configured correctly.
Restore Structure History	If selected, structures are imported along with their history (if it is present in the backup file). If not selected, structures will have no history.
Restore User Favorites	If selected, the plugin will try to restore "favorite" marks made by users for the selected structures.
Restore Views	If selected, all views from the backup files will be restored. If there's a conflict and a view with a given ID already exists, Structure will first verify if the view being restored is different from the one in the system. If it is, Structure will restore the backup view as a new view with a different ID.
Restore View Settings for Structures	If selected, view settings(see page 541) for the selected structures will be restored.
Restore Configuration	If selected, the global app permissions(see page 954) will be restored. This includes the list of projects, for which Structure is enabled, permissions to use or create structures, permissions to use Automation, etc.
Restore Dark Features	If selected, the dark features settings(see page 974) will be restored.

6. Click **Restore Selected Structures**.
7. You will see the Process Status page. Once the restore process is completed, the **Show Result** button will take you to the result page, where you'll be able to see the result and any warning messages.

After structures have been migrated, open the **Structure | Manage Structure** page to verify that your structures were restored successfully.

⚠ As of version 3.3, Migrate Structure does not support Structure.Testy or Structure.Pages data.

9.10.4.1 Owners and Permissions

When migrating structures, the following ownerships and permissions are included in the migration:

- Structure owners
- View owners
- Structure permissions
- "Apply Permissions" rules, if the structures they refer to are migrated
- When a permission rule cannot be migrated, Structure logs a warning and skips this specific rule

⚠ Backups created in a version of Structure prior to 6.2 do not contain usernames and project role names. When trying to migrate from an older backup file, ownerships and permissions that require this information will not be included.

9.11 Automatic Structure Maintenance

9.11.1 Automatic Structure Maintenance

Automatic Structure maintenance runs daily and performs Structure backup and database optimization. The optimization removes stale data from the database and may improve general Jira responsiveness.

⚠ Automatic maintenance can be run only with a valid Structure license.

To configure automatic Structure maintenance:

1. Navigate to **Administration | Structure | Maintenance**
2. Click **Configure Scheduled Maintenance**
3. If scheduled maintenance is disabled, click **Enable scheduled maintenance**
4. Select the maintenance schedule
5. Select tasks that scheduled maintenance should run
6. Configure additional task parameters, if any
7. Click **Apply**

⚠ By default, scheduled maintenance is enabled and set to run daily at 3 AM.

9.11.2 Maintenance Schedule

You have several options to specify a maintenance schedule:

1. Run every day at a specified time

   ✓ The time is specified in the server's time zone, displayed near the time fields.

2. Run based on a crontab schedule
   Your schedule should follow standard crontab formatting. The schedule is a list of five, single-space-separated fields, representing: minute, hour, day, month, weekday. Each field can be a value, list of values or

range. Month and weekday names can be given as the first three letters of the English names. Among numbers and month/weekday names, the following symbols can be used:

- Asterisk (*) is used to include every value in that field. For example, * in the month field indicates every month of the year.
- Question mark (?) is used instead of '*' for leaving either day-of-month or day-of-week blank.
- Comma (,) is used to separate items of a list. For example, using "MON,WED,FRI" in the 5th field (day of week) means Mondays, Wednesdays and Fridays.
- Hyphen (-) defines a range between two specified values. For example, 2000–2010 indicates every year between 2000 and 2010, inclusive.
- Slash (/) can be combined with a * to specify step values. For example, */5 in the minutes field indicates every 5 minutes.

Schedule examples:

- 0 * * * * = the top of every hour of every day.
- */10 * * * * = every ten minutes.
- 0 8-10 * * * = 8, 9 and 10 o'clock of every day.
- 0 6,19 * * * = 6:00 AM and 7:00 PM every day.
- 0/30 8-10 * * * = 8:00, 8:30, 9:00, 9:30, 10:00 and 10:30 every day.
- 0 9-17 * * MON-FRI = on the hour nine-to-five weekdays.
- 0 0 25 12 ? = every Christmas Day at midnight.

9.11.3 Maintenance Tasks

Backup Structure data	Creates a backup of the Structure database in the `export` sub-directory under Jira home. Parameters: • Include history – if checked, full structure change history will be included in the backup. If you have a lot of changes in structures, this setting may cause the backup to take some time, and the backup file to be large. If you don't need a history of structure changes, it is advised to turn this option off. It is recommended that you maintain separate Structure backups, even though Structure data is backed up with Jira's normal backup, because this will allow you to restore from that data without rolling back changes in Jira.
Delete old backups	A backup is considered old if it is not among *X* latest backups (*X* is specified by the first parameter of this task) **and** it was made earlier than *Y* days ago (*Y* is specified by the second parameter). This task removes all such backups made by the Backup task. Parameters: • Always keep *X* latest backups • Always keep backups made during the last *Y* days
Optimize favorites	If a user marks a structure as their favorite(see page 537), Structure will keep this mark, even if the user is later deleted from Jira, and the popularity number of the structure(see page 537) will still accounts for that user. This task removes marks made by users no longer in Jira and recounts the structure popularity.

Optimize structures	If an issue is added to a structure and then deleted from Jira, the structure will still contain a reference to this issue (although it will not display it). This task removes references to deleted issues and other items that have become permanently unavailable.
Optimize view settings	If a view(see page 966) is deleted, some structure view settings(see page 541) may still reference it, and a blank view named **? (Unknown View)** will be shown in its place. This task removes references to deleted views.
Optimize synchronizers	This removes an data related to synchronizers of a deleted structure.
Delete old synchronizer audit log records	This removes old records from the Synchronizer Audit Log(see page 966), clearing up space in the database. Parameters: • Keep records for the last *X* days. If you set X to 0, the maintenance procedure will remove all records.
Reindex change history	Currently does nothing. This task has remained as an option since Structure 2. Its purpose may be restored in a later version of Structure.
Optimize structure perspectives	Removes old perspectives(see page 97) that haven't been used by anyone for a certain amount of time. Parameters: • Delete perspectives that were not used during the last *X* days
Reindex structures	Clears and recalculates issue-to-structure index, used to define which structures contain a specific issue. (Issues added with Generators(see page 140) are not counted.)

Delete old change history	The task removes old records from the change history. A history record is considered old if the change was made earlier than X days ago (X is specified by the first parameter) **and** it is not among the Y latest history records for the structure where the change was made (*Y* is specified by the second parameter). Parameters: • Always keep change history for the last X days • Always keep the Y latest changes per structure

9.11.4 Running Maintenance Tasks Manually

You can run specific maintenance tasks at any time.

To run maintenance manually:

1. Navigate to **Administration | Structure | Maintenance**
2. Navigate to the **Run Maintenance Now** section
3. Select tasks to run
4. Configure additional task parameters, if any
5. Click **Run Maintenance Now**

Running maintenance manually does not affect the automatic maintenance settings or schedule.

9.12 Workflow Integration

9.12.1 Structure Workflow Validator

Structure Plugin adds a new workflow transition validator[631] to JIRA. This validator blocks the transition if the issue doesn't match an S-JQL query(see page 428). For example, it can be used to prevent an issue from being resolved if the issue has some unresolved sub-issues in a structure.

Adding an S-JQL condition for the issue creation transition will result in the S-JQL condition always evaluating to false. The validation check is done before the issue is inserted into the structure, so the S-JQL check won't find the issue.

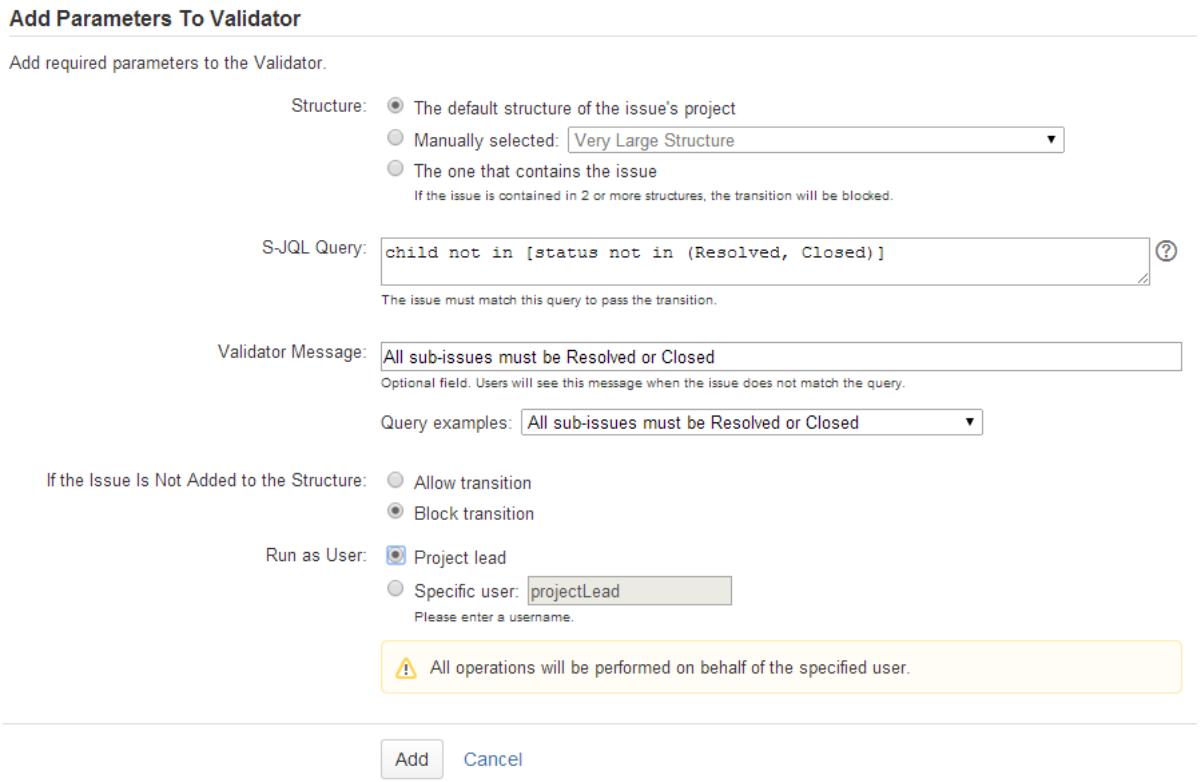
To add the Structure validator to a workflow:

1. Create a draft of the workflow and open the **Add Validator to Transition** dialog. (For more information, please refer to the Jira documentation[632].)

631 https://confluence.atlassian.com/display/JIRA/Advanced+workflow+configuration#Advancedworkflowconfiguration-Whatareconditions,validators,andpostfunctions?

632 https://confluence.atlassian.com/display/JIRA/Advanced+workflow+configuration#Advancedworkflowconfiguration-validatorAddingavalidator

2. Select the **Issue Matches Structure Query** validator. A configuration window will open:



3. In the **Structure** field, specify how the validator should select the structure to check. It can either be a manually-selected structure, or it may depend on the issue being checked (the default structure of the issue's project or the structure that contains the issue).

> ⚠ In Structure 3, the option to pick a structure that contains the issue being validated is no longer available.

4. In the **S-JQL Query** field, enter the S-JQL query that the issue should match in order to pass the transition.

> ✓ You can use one of the examples provided with the form. Just select an example in the **Query Examples** selector, and the corresponding query will be copied into the **S-JQL Query** field.

5. In the **Validator Message** field, enter an explanation message that users will see if their transitions are blocked by the validator.
6. In the **If the Issue Is Not Added to the Structure** field, select whether the transition should be blocked or allowed if the issue is not contained in the checked structure. (Or if the issue does not belong to any structure, in case automatic structure selection is chosen.)
7. In the **Run as User** field, select on behalf of which user the validator should run. It can either be a manually-selected user, or it may be the lead of the project of the issue that is being checked.

> ⚠ Running on behalf of a user means that the validator will only see issues and structures that are accessible to the specified user. The result of the validator check will depend on the permissions of the specified user and will not depend on permissions of the user who performs the transition.

9.12.2 Structure Workflow Condition

Structure Plugin also comes with the Structure condition that is similar to the Structure validator. Using Structure condition may significantly increase the load on server, so this condition is not available by default.

To make Structure Workflow Condition available, enable the **structure-workflow-condition** module of the Structure plugin via **Administration | Add-ons | Manage Add-ons** page. For instructions, please see Universal Plugin Manager documentation[633].

> Checking S-JQL condition may involve querying other issues in the checked structures, and in case the structure is large, this may take considerable time – yet within reason if it is done occasionally.
>
> However, workflow conditions are checked every time a user opens an issue details page, in order to decide which transitions to show. If you have hundreds of active users and thousands of issues in a structure, this may easily degrade server performance.
>
> Use your own best judgement.

9.13 Running Structure on Jira Data Center

Structure is a Data Center approved app(see page 679).

The following articles provide additional information, specific to the Data Center editions.

- Archived Projects and Structure(see page 971)

To learn more about the Data Center approved app program, see our Data Center Approved Apps FAQ(see page 679).

9.13.1 Archived Projects and Structure

Administrators can archive projects and/or issues in Jira Data Center.

When this happens, the archived issues become read-only and can only be accessed through a direct link. They are removed from the Lucene index and will no longer be part of search results, even if they match the search criteria.

9.13.1.1 Archived Issues in Structure

> Archived issues will not be visible in Structure – even if they were added manually, and even in the archived structures.

When you archive a project, check for structures that have issues from that project. Contact the users to see if they are okay with the archived issues disappearing from the structures.

633 https://confluence.atlassian.com/display/UPM/Viewing+installed+add-ons#Viewinginstalledadd-ons-AboutAdd-onModules

9.13.1.2 Restoring an Archived Project

When you restore a project from archive, you run a project reindex. After the reindex has finished, the issues should reappear in Structure.

Since Structure has internal caching for which issues are visible to which users, it might take a few minutes after the reindex completes for issues to reappear in Structure. To force the internal caches to clear and see the restored issues in structures immediately, use the "force reload" action from your browser (Shift+Reload).

9.14 Anonymous Usage Statistics

Please enable anonymous usage statistics, as it helps the developers better understand how Structure is used, prioritize improvement requests and build a better product. No Jira content or personally identifiable data is collected.

When anonymous usage statistics is enabled, Structure periodically sends some data from the Jira instance to ALM Works.

The data consists of anonymized information related to the usage of Structure, such as an invocation count of each structure widget action (structure history is toggled 56.3 times a day on average, issues are pasted 30.7 times a day on average, etc.).

Here is a sample report that is sent to ALM Works: Statistics Sample[634]

9.14.1 Viewing Current Statistics

Jira administrators can always review the data that is about to be sent. To view the data:

1. Navigate to **Administration | Structure | Support**
2. Click **View Current Statistics**

9.14.2 Turning Anonymous Usage Statistics On and Off

To enable or disable Anonymous Usage Statistics:

1. Navigate to **Administration | Structure | Support**
2. Check or uncheck the **Send anonymous usage statistics** checkbox
3. Click **Apply**

The information is collected in accordance with EULA[635] and our privacy policy[636].

634 https://wiki.almworks.com/download/attachments/200605761/statisitics.txt?api=v2&modificationDate=1712580882000&version=1
635 http://almworks.com/EULA-Structure.pdf
636 http://almworks.com/company/legal/privacy.html

9.14.3 Structure Usage Statistics

Recent statistics can be viewed at any time.

1. Go to **Administration** | **Structure** | **Support** and scroll to the **Anonymous Usage Statistics** section near the bottom of the page.
2. Click **View Current Statistics** to see the Structure usage data recorded for the past 30 days.

> Usage statistics are always recorded, but are only sent to to us if the '**Send anonymous usage statistics**' option is enabled. Sharing this anonymous data with us helps us improve our product, but if you do not wish to, simply uncheck this option.

> Jira administrator rights are required to access these statistics.

The log shows statistics related to different actions in structures - number of created items, description of existing generators, number of clicks made on specific buttons, etc.

Among this data, there is:

- The *activeUser* parameter, which shows the number of users who performed meaningful actions in any structure - created new items, added/removed/moved items, etc. Users who only viewed structures, without changing anything, are not counted.
- The *structureCount* parameter, which shows the total number of structures in the system.

9.15 Turning Off Optional Features

Some features in Structure are designed as modules and can be safely turned off. This can be done to remove unnecessary functionality or limit the exposure to Structure for specific users.

> If your aim is to limit the exposure to Structure, consider restricting permissions to specific groups of users - see Structure Best Practices(see page 921).

> While it is easy to disable many Structure modules, we don't recommend touching any modules except those listed in this article to ensure stability of Structure and your Jira application.

To turn off a module:

1. Open the Add-on Manager by navigating to **Administration | Add-ons | Manage Add-ons**.
2. Locate the Structure add-on and expand its row.
3. Click the link that looks like the following: "309 of 310 modules enabled." (Numbers may vary.)
4. Use the Search feature of your browser to find the module by its name (provided below.)
5. Click the **Disable** button to the right of the module name.

You can always turn the feature on later by clicking the **Enable** button.

Feature	Module names	Effect of disabling this module
Activity Streams	`Structure (structure-activity-provider)`	The Activity Streams provider and Structure-related updates are removed from the following places: • Activity Stream gadgets(see page 562), • Activity tab on the issue page • Activity tab on the user page • Activity tab on the project page
Structure on the Issue Page(see page 90)	`web-resource:Issue Page Decorator (adjustIssue)` `issue-page:Structure on Issue Page (structuremodule)`	The Structure section is removed from the issue page.
Structure on Agile Boards(see page 96)	`web-panel:GreenHopper tab (greenhopper-tab)`	The Structure tab is removed from the issue details panel on the Agile board.

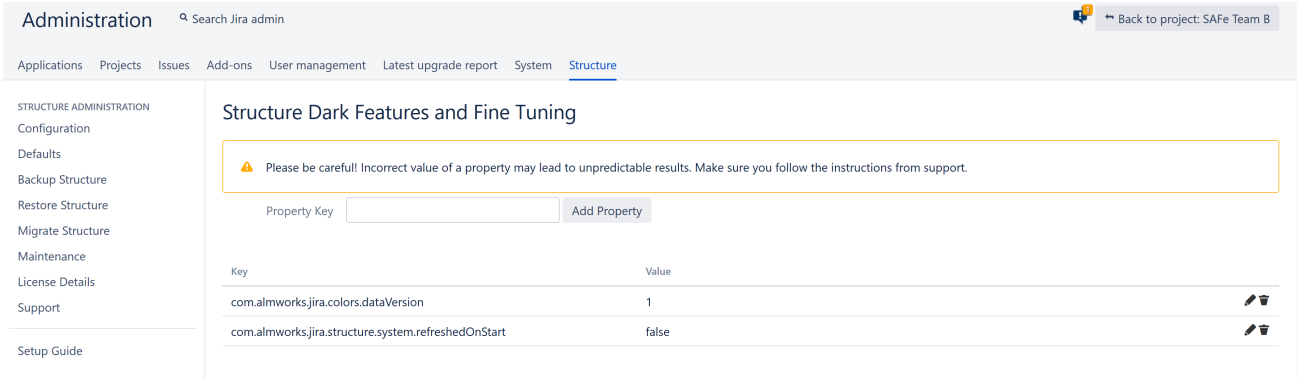
9.16 Advanced Configuration and Dark Features

Certain advanced aspects of Structure's behavior might not have dedicated configuration pages, being controlled by application properties or system properties instead. Additionally, some features and behaviors are usually hidden from users by default. These dark features can be turned on for an instance by the Jira administrator.

9.16.1 Setting Application Properties

The easiest way to add and manage custom Structure properties and dark features is to use the Structure Dark Features and Fine Tuning interface.

- To add a new custom property or dark feature, enter the appropriate Property Key (see below for a list of available keys) and click **Add Property**.
- Once the key is added to your properties list, you can adjust its value by clicking the edit icon (pencil).
- To remove a custom property, click the trash icon.
- Once you have made all necessary updates, Structure must be restarted(see page 982) to apply the changes.



To access the interface, you must have Jira Administration permissions and enter the interface location directly into your browser: https://YOUR_JIRA_ADDRESS/secure/admin/StructureDarkFeatures.jspa

9.16.1.1 Guidelines for Editing Properties

- When an invalid property value is entered in the table, the default value is applied.
- Spaces are not trimmed, and may result in an invalid value.
- When you delete a property from the admin table, it's property value is set to the default value:
    - If the property was added with our admin interface, the value is set to empty value and the property is removed from the table after a page refresh.
    - If you set the value to empty (without deleting the property), the property will not be removed.
- Structure must be restarted(see page 982) to apply changes.

9.16.2 Setting System Properties

You can set System properties during Startup or using Script Runner.

> ⚠ Both of the following methods can also be used to set Structure properties; but we recommend using the Structure Dark Features and Fine Tuning interface as described above.

9.16.2.1 On Startup

You can set System properties using the `-D` Jira startup option, for example:

```
-Dstructure.sync.guard.email.admin.cycles=5
```

Configuring Jira startup options is described in this article[637]. You will need to restart Jira for the properties to take effect.

9.16.2.2 With Script Runner

You can also set system properties using the Script Runner[638] add-on.

1. Install Script Runner.
2. Go to **Administration | Add-Ons | Script Runner | Script Console**.
3. Select **Groovy** as the Script Engine.
4. Enter the following code into the Script text box, adjust property name and value as needed, and click **Run Now**.

```
System.setProperty("structure.sync.guard.email.admin.cycles", "5")
```

The changes take effect after you restart Structure(see page 982), but the properties will be reset to their default values when you restart Jira. In some cases, for settings to take effect you have to reinstall Structure.

9.16.3 Available Dark Features

9.16.3.1 Structure size limit

Property	Default	Explanation
`com.almworks.jira.structure.AOBasedStructureManager.forestSizeLimit`	`100000`	The maximum number of rows that one structure can contain.

637 https://confluence.atlassian.com/display/JIRA/Setting+Properties+and+Options+on+Startup
638 https://marketplace.atlassian.com/plugins/com.onresolve.jira.groovy.groovyrunner

9.16.3.2 Structure Automation limits

Property	Default	Explanation
`structure.gfs.generationTimeHardLimit`	`600`	The maximum amount of time that can be spent for Structure generation (in seconds).

9.16.3.3 Automation Defaults

Property	Default	Explanation
`structure.generator.defaults.disableUpdates`	`false`	When adding generators: • If set to "false" (default) - the "allow changes" box is initially checked. • If set to "true" - the "allow changes" box is initially unchecked.

9.16.3.4 Manual adjustments

Property	Default	Explanation
`structure.gfs.manualAdjustments.enable`	`true`	Setting this property to false will disable manual adjustments for the entire Jira Instance. All adjustment-related UI elements and controls will disappear. Existing manual adjustments will be kept in the database, but will not be applied.
`structure.gfs.manualAdjustments.maxAdjustmentsPerStructure`	`20000`	The maximum number of manual adjustments per one structure. When this limit is reached, adding new manual adjustments will be impossible. If you reduce this limit, you may have to remove all manual adjustments for the structures that exceed it.
`structure.gfs.manualAdjustments.maxAdjustmentsPerAction`	`200`	The maximum number of manual adjustments per one user action. If this limit is exceeded, the action will be aborted without making any changes.

9.16.3.5 Hidden Issue Links

Property	Default	Explanation
`structure.feature.hiddenLinks.enabled`	`false`	Set to true to enable support for hidden issue links.

9.16.3.6 Index Consistency Checks

Property	Default	Explanation
`structure.indexConsistencyChecker.disabled`	`false`	Set to true to disable periodical checks of Lucene index consistency.

9.16.3.7 Alternative initial values for project/type when creating an issue in dialog

Normally, when the user creates new issues through dialog, Structure remembers the selected project and issue type and offers those the next time by default. This dark feature enables a different algorithm, which used to work in a previous version of Structure: the initial project and issue type are taken from the issue that was focused when "+Create" was pressed.

System property	`structure.feature.altInitialValuesInDialog`
Options to add in setenv.sh / setenv.bat	`-Dstructure.feature.altInitialValuesInDialog=true`
Internal feature name	`altInitialValuesInDialog`
Introduced in version	2.11.0

9.16.3.8 Managing Permissions

Property	Default	Explanation
`structure.permissions.allowAllUserGroups`	`false`	By default, users can only manage permissions for groups they are a part of. Setting this property to "true" allows users to manage any group, whether they belong to that group or not.
`structure.permissions.jiraAdminHasAllPermissions`	`true`	By default, Jira admins are granted all Structure global permissions(see page 954), Control(see page 543) access to all structures, and Manage(see page 513) access to all views. If this property is set to "false," all global permissions and access restrictions apply to Jira admins, just like for regular users.

9.16.3.9 Resolved icon (green tick)

Property	Default	Explanation
`structure.doneAttribute.byResolution`	`false`	false - the "Resolved icon" shown when the Resolution field of an issue is non-empty true - the "Resolved icon" is shown when the issue is in a Done status category (StatusCategory.COMPLETE)

9.16.3.10 Show Archived Timesheets by Tempo Accounts in the Tempo Worklogged column

Property	Default	Explanation
`structure.tempo.showArchivedAccounts`	`false`	false - archived accounts are not shown true - archived accounts are available in the Tempo Worklogged drop down

9.16.3.11 Structure Export Timeout

For large, complex structures, it may take a long time to capture all the details of a structure required for exporting. In order to reduce the amount of time and resources required for export, after a set time Structure will attempt to

use a cached version of the structure for the export. If this is not possible, it will continue trying to compile live data for a set time before the export fails.

Property	Default	Explanation
`structure.export.trySnapshotTimeout`	`120`	In seconds, the amount of time Structure will wait before attempting to export a cached version of the structure. Note: when a cached version is exported, a warning is added to the export.
structure.export.forestLockTimeout	`600`	In seconds, the total time allotted before the export fails.

⚠ Minimum time is 60 or 90 seconds, and it can take up to 60 seconds longer than the configured timeout, because lock time is measured in 30s increments (by default), and there are 2 or 3 loops where the exporter waits, depending on the attributes and forest spec.

9.16.3.12 Time in Status - Refresh Period

Property	Default	Explanation
`structure.timeinstatus.refreshPeriod`	`3600000`	Sets update period for Time in Status columns. Value is in milliseconds. Time is Status is updated any time a status change occurs in Jira or once per hour if status remains unchanged. This option allows you to update the Time in Status more or less often, when issues remain in the same status.

9.16.3.13 Time Tracking on the Issue Page and Issue Details Page

When viewing an Issue page in Jira or the Issue Details page in Structure, the Time Tracking section provides up-to-date time tracking details, and allows you to include time tracking details for sub-issues within the calculations. If these pages are taking too long to load, disabling this feature may help.

Property	Default	Explanation
`structure.issuePage.showTimeTracking`	`true`	false - disables the Structure Time Tracking section on the Issue page and Issue Details page.

9.16.3.14 Link-related changes made by Generators or Synchronizers

By default, when a generator (see Generators(see page 140)) or Synchronizer creates, deletes or changes a link, the change appears immediately on an Issue Page, but it is not immediately indexed in Jira. Additionally, the change is not written to history, so it is not possible to review the changes on an issue's History tab.

This setup prevents link changes from negatively impacting system performance (a particular risk when bulk changes are made to issue links); however, there may be times when these types of changes need to be indexed immediately and/or appear in an issue's history. To enable this, simply update the following properties.

Property	Default	Explanation
`structure.bulkLinkProcessor.useLinkManager`	`false`	true - enables the immediate re-indexing after every change, writes changes in the history and creates a Structure-side event for each link update (enabling other apps to see that Structure made the change).
structure.sync.links.force.reindex	`false`	true - enables immediate indexing.

9.16.3.15 Synchronizers

Synchronization(see page 983) lets you keep Structure issue hierarchy in sync with some other issue properties. This dark feature is disabled by default.

Property	Default	Explanation
`structure.feature.synchronizers.enabled`	`false`	Set to true to enable Synchronizers within Structure.

Synchronizer Cycle Guard

The cycle guard(see page 991) is a component that detects conflicting synchronizers and prevents them from cycling forever, overriding each other's changes. The table below describes the system properties that control the cycle guard.

Property	Default	Explanation
`structure.sync.guard.disable`	`false`	Set to `true` to disable the cycle guard. Conflicting synchronizers will not be prevented from running forever. **Not recommended.**

Property	Default	Explanation
`structure.sync.guard.maxAutosyncsWithoutUserChanges`	`10`	The maximum number of times that a synchronizer is allowed to run, processing the changes generated by another synchronizer. If this limit is exceeded, the two synchronizers are considered to be in conflict.
`structure.sync.guard.stop.disable`	`false`	If `true`, conflicting synchronizers will not be disabled automatically. The cycling may repeat after a user-generated change.
`structure.sync.guard.email.owner.disable`	`false`	If `true`, the cycle guard will never send e-mail notifications to synchronizer owners.
`structure.sync.guard.email.admin.disable`	`false`	If `true`, the cycle guard will never send e-mail notifications to Jira administrators.
`structure.sync.guard.email.admin.cycles`	`10`	The minimum number of times a cycle must be detected for a synchronizer before an e-mail notification about that synchronizer is sent to Jira administrators. The counter is reset when a synchronizer is automatically disabled, so if this number is greater than 1 and automatic disabling is on, the administrators will not be notified.

9.16.4 Restart Structure

After applying a dark feature, it may be necessary to restart Structure for the change to take effect. To restart Structure:

1. Go to **Administration | Manage apps**
2. On the sidebar, click **Manage apps**
3. Locate **Structure**
4. Click **Disable** (when prompted, skip sending feedback)
5. Click **Enable**

9.16.5 Synchronization

Dark Feature - Synchronizers are hidden by default

If there are no synchronizers installed on your system, this feature will now be hidden from users. To enable synchronizers, set **structure.feature.synchronizers.enabled** system property to **true**. See Advanced Configuration and Dark Features(see page 974) for more information.

If you currently have synchronizers installed, the feature will remain visible.

Synchronization lets you keep a structure issue hierarchy in sync with some other issue properties. For example, you can enforce the rule that Jira sub-tasks should always be placed under their parent in the structure, or that there should be an issue link from parent issue to each sub-issue.

Synchronization can also be run once to perform a one-time update of the structure (Import) or one-time update of the issues based on the structure (Export).

Synchronization is an extendable system that allows Jira add-ons to provide their own synchronizers. The following synchronizers are supplied with Structure:

- Sub-Tasks Synchronizer(see page 993) places Jira sub-tasks under their parent issues in the structure.
- Links Synchronizer(see page 997) makes sure sub-issues are linked to their parent issues with a specific link type. It can also be used to reconstruct structure from links.
- Filter Synchronizer(see page 993) populates structure with issues that pass a saved filter. It can also be used to remove issues from a structure when they no longer pass the filter.
- Agile Synchronizer(see page 1000) works to sync the Agile ranking of issues with their position in the structure and to make it easier to assign stories to epics using Structure.
- Status Rollup Synchronizer(see page 1003) propagates issue status upwards. For example, it can make parent issue Resolved if all sub-issue are resolved.

One-time synchronization works when you run Export(see page 985) or Import(see page 984), or when you run a Resync(see page 989). Automatic synchronization runs in the background and listens for updates in the structure and beyond.

Please be careful: synchronization may cause massive changes to issues. For example, if you install the Agile synchronizer and then add issues to the structure in some random order, those issues' ranks will be changed according to that order almost immediately! We highly recommend that you maintain daily backups and carefully review how a synchronizer works before installing it.

In order to install a synchronizer, you need to have **Control** permissions for a structure and have necessary permissions for the Jira issues.

Note that you also need to have special permission to **Control Synchronizers**. If there is a warning message above the feature description, please contact your *Jira Administrator* for assistance.

Anonymous users cannot install synchronizers or use Export/Import, even if they are granted Control permissions.

9.16.5.1 Import Synchronization

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

When you run an **Import** synchronization, you get a set of issues which should be in the structure and/or information on how they should be arranged in a hierarchical list. This information is then applied to an existing structure.

For example, you can use a Filter Synchronizer(see page 993) to add All Open Issues to a structure (or the results of whatever Saved Filter you have), or you could use the Agile Synchronizer(see page 1000) to rearrange issues in the structure according to their rank and epic.

> To run Import, you must have **Control** permissions on the structure and additional permissions may be required by a specific synchronizer.

To import hierarchy into a structure:

1. Open the Manage Structures(see page 532) page using the top navigation Structure menu.
2. Select the structure you'd like to import into and click the **Import** link.

   > If you don't see the **Import** link in the Operations column, you may not have Control permissions for this structure.

3. Select a synchronizer from the drop-down list and configure the import parameters.

   > If there are no synchronizers in the drop-down list, either none are currently installed or none of the installed synchronizers support import into a structure.

4. Enter the synchronizer parameters. Each synchronizer has its own parameters, so please refer to the specific synchronizer documentation(see page 992).
5. Click **Run Import**. When you start the import, the synchronizer will analyze the existing data and possibly update the entire structure.
6. After you click Run Import and confirm the operation, a job status page is shown. When the job is marked as Finished, the synchronization is done, and you can view the results by opening the affected structure.

> When import is run, it runs under your user name and with your permissions. If you don't have enough permissions to read values somewhere else or to view issues you'd like to import, you may not see the expected result.

Import(see page 984) and Export(see page 984) are actually a one-time Resync(see page 989). Export is resync from Structure and import is resync into Structure. If you need to run export or import periodically, you can set up a synchronizer with all the parameters but without enabling it - so no synchronization happens in the background. When you need to export or import, you can open Synchronization Settings page for the structure and run Resync. Just make sure you've selected the correct Resync direction!

9.16.5.2 Export Synchronization

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

When you run an **Export** synchronization, you use the hierarchy from a structure to create, update or delete issue attributes or features, based on the hierarchy that a specific synchronizer provides.

For example, you can use the Links Synchronizer(see page 997) to create issue links between sub-issues and their parents.

To run Export, you must have **Control** permissions for the Structure, and you will likely need some additional permissions, depending on which synchronizer you're going to use. For example, you have to have *Link Issues* permission when working with the Links synchronizer.

To export hierarchy from a structure:

1. Open the Manage Structures(see page 532) page using the top navigation Structure menu.
2. Select the structure you'd like to export from and click the **Export** link.

   If you don't see the **Export** link in the **Operations** column, you may not have **Control** permissions on this structure.
3. Select a synchronizer from the drop-down list and configure the export parameters.

   If there are no synchronizers in the drop-down list, either none are currently installed or none of the installed synchronizers support exporting from a structure.
4. Enter the synchronizer parameters. Each synchronizer has its own parameters, so please refer to the specific synchronizer documentation(see page 992).
5. Click **Run Export**. When you start the export, the synchronizer will read the current structure and apply it to whatever it syncs with.
6. After you have clicked Run Export and confirmed the operation, a job status page will be present. When the job is marked *Finished*, the synchronization is done and you can inspect the results.

When export is run, it runs under your user name and with your permissions. If you don't have enough permissions to make a certain change in Jira, the synchronizer will skip that change (a warning will be printed out in the server logs).

Import(see page 985) and Export(see page 985) are actually a one-time Resync(see page 989). Export is resync from Structure and import is resync into Structure. If you need to run export or import periodically, you can set up a synchronizer with all the parameters but without enabling it - so no synchronization happens in the background. When you need to export or import, you can open Synchronization Settings page for the structure and run Resync. Just make sure you've selected the correct Resync direction!

9.16.5.3 Installing a Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

When you install a synchronizer on a structure, you make this structure automatically sync with something else (as opposed to doing a one-time import(see page 984) or export(see page 985)).

Here are a couple of examples of how that might work:

- With a Links Synchronizer(see page 997) installed and enabled, any changes someone makes to the structure will cause issue links to be created or deleted in Jira to match those changes.
- With a Filter Synchronizer(see page 993) in *Add* mode installed and enabled, when someone creates or changes an issue in Jira, if that issue now passes the selected saved filter, it will be automatically added to the structure.

When you install a synchronizer, you define its parameters. These parameters can be edited(see page 987) later.

To install a new synchronizer:

1. Open the **Manage Structure** page using the top navigation Structure menu.
2. Find the structure you'd like to sync. The **Sync With** column shows currently-installed synchronizers. Click the **Settings** link in that column.

   If you don't see the **Settings** link in the **Sync With** column, you may not have **Control** permissions for this structure.
3. The synchronization settings page shows detailed information about each installed synchronizer and lets you work with them. To proceed with the installation of a new synchronizer, select the type of synchronization and click **Configure and Install Synchronizer**.
4. Enter the synchronization parameters. Each synchronizer has its own parameters, please refer to the specific synchronizer documentation(see page 992).
5. Press the **Create** button to install the synchronizer. Though installed, the synchronizer is not enabled yet.
   a. Before synchronization is enabled, we recommend running Resync(see page 989) to sync the current state of the structure and Jira. To do so, press the **Resync and Enable** button after the synchronization is installed, or later use the same link on the synchronization settings page.
   b. If you need to enable synchronization without resyncing first, press **Enable without Resyncing**.
   c. You can enable and resync the synchronizer later from the synchronization settings page. Press **Done** if you don't need to enable the newly installed synchronizer now.

After a synchronizer is installed, it's not working yet - it must be **Enabled** to start monitoring the changes.

Import(see page 986) and Export(see page 986) are actually a one-time Resync(see page 989). Export is resync from Structure and import is resync into Structure. If you need to run export or import periodically, you can set up a synchronizer with all the parameters but without enabling it - so no synchronization happens in the background. When you need to export or import, you can open Synchronization Settings page for the structure and run Resync. Just make sure you've selected the correct Resync direction!

9.16.5.4 Modifying Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

You can change the parameters of a synchronizer to alter how it works.

If the synchronizer is enabled (and working in the background), it needs to be turned off(see page 988) before you can make any changes, because changing a synchronizer's parameters may completely change the result of synchronization.

To edit a synchronizer:

1. Open the **Manage Structure** page using the top navigation Structure menu.
2. Find the structure with the synchronizer you'd like to edit. Click on the **Settings** link in the **Sync With** column.

   If you don't see the **Settings** link in the **Sync With** column, you may not have **Control** permissions on this structure.
3. Find the synchronizer you'd like to edit. Click the **Edit** link or the **Disable and Edit** link in the **Operations** column. The synchronizer will be automatically disabled.

   If you don't see the **Edit** link or the **Disable and Edit** link in the **Operations** column, the synchronizer is probably provided by a third-party plugin and does not support editing.
4. Set the new synchronizer parameters. Also, if you are a **Jira Administrator** and not the synchronizer owner, choose if you want to become the new synchronizer owner.
5. Press the **Apply** button. This will update the synchronizer parameters and make you the new owner of the synchronizer (unless you chose not to do so in the previous step). However, the synchronizer is not enabled yet:
   a. Before synchronization is enabled, we recommend running Resync(see page 989) to sync the current state of the structure and Jira. To do so, press the **Resync and Enable** button after the synchronization is installed, or later use the same link on the synchronization settings page.

b. If you need to enable synchronization without resyncing first, press **Enable without Resyncing**.
c. You can enable and resync the synchronizer later from the synchronization settings page. Press **Done** if you don't need to enable the updated synchronizer now.

9.16.5.5 Removing Synchronizer

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

You can remove an installed synchronizer at any time if you have **Control** permissions for the structure.

1. Open the **Manage Structure** page using the top navigation Structure menu.
2. Find the structure you need to remove a synchronizer from. You can look at the *Sync With* column to see which synchronizers are installed in a structure.
3. Click **Settings** link for the selected structure.
4. Find the synchronizer in the list and use the **Delete** link to remove it.

> If the synchronizer is currently performing an incremental sync or resync, it will be allowed to finish.

9.16.5.6 Turning Synchronizer On and Off

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

A synchronizer can be in one of the following states:

- **Disabled** - the background incremental synchronization is not running. You can run resync(see page 989) to do a one-time full sync.
- **Enabled** - the background incremental synchronization is running. When a change is detected, synchronization applies the change to the other part of the synchronous link as soon as possible, typically within several seconds.
- **Not Available** - the synchronizer is installed but it cannot run any synchronization. The possible reasons are changes in Jira configuration or lack of permissions from the user.

> Synchronizers are disabled by default. After installation(see page 986), they must be enabled, typically with a resync(see page 989).

Turning a Synchronizer On

To **enable** an inactive synchronizer:

1. Open the synchronization settings page for the structure.

2. Find the synchronizer and click the **Enable** link.
3. Alternatively, you can click **Resync and Enable** to resync(see page 989) and enable the synchronizer immediately after the resync finishes.

Turning a Synchronizer Off

To **disable** an active synchronizer:

1. Open the synchronization settings page for the structure.
2. Find the synchronizer and click the **Disable** link.

> If the synchronizer is currently running a sync, it will be allowed to finish.

9.16.5.7 Running Resync

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

A resync, or full resynchronization, is a one-time process activated manually by the user to sync current state of the structure and Jira.

> Resync scans all issues that may be affected by the synchronizer - contrary to the incremental synchronization, which inspects only issues that have been changed.

Resync Direction

When running a Resync, you need to choose it's direction.

Resync can be run:

- *from Structure* - Issues in Jira will be updated based on the structure hierarchy. This is what happens when you export from a structure(see page 985).
- *into Structure* - The structure hierarchy is updated (issues will be moved, added or removed), based on the current state of the issues in Jira. This is what happens when you import into a structure(see page 984).

A synchronizer may support resyncing in only in one direction. For example, the Saved Filter synchronizer, which adds issues from a saved filter result, can only sync *into* Structure.

> The incremental (background) sync tries to apply changes on *both* sides to the other side, if possible, depending on where the change has happened. For example, with the Agile synchronizer, if you change the rank (issue position in the backlog on the Planning Board), its position in the structure is also changed; and if you change the position in the structure, it's rank is changed in Jira.

Running Resync

To run a resync:

1. Open the **Manage Structure** page using the top navigation Structure
2. Click the **Settings** link in the *Sync With* column for your structure
3. On the synchronization settings page, find the synchronizer you'd like to Resync, and either
    a. Click **Resync**
    b. Click **Resync and Enable** if the synchronizer is disabled and you'd like to enable it immediately after the Resync finishes
4. Select a direction for the Resync. See *Resync Direction* above.
5. Click **Start Resync**.

    Resyncing in a wrong direction may lead to data loss! Please make sure you understand how each direction works and confirm running the resync when a confirmation dialog appears.

6. The job status page that appears will tell you when the Resync has finished.

If the synchronizer is currently running an incremental synchronization, the resync will wait until it finishes.

9.16.5.8 Synchronization and Permissions

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

When sync runs, the updates will be made on behalf of the user who installed the synchronizer! Any change that a synchronizer makes when reconciling data between Structure and Jira is performed on behalf of the user who created the synchronizer, not the user who ran the synchronizer.

This is really important to understand. Consider the following settings:

- You create a Structure and set up structure permissions(see page 543) so that anyone can edit the structure.
- You have Link Issues permissions on a project, and you install a Links synchronizer to have child issues linked to their parent issue.

Now anyone can edit the structure - add issues, remove issues and rearrange the issues in the structure. **Every change to the structure will lead to adding and removing links between the affected issues on your behalf - even if the user who changes the structure does not have Link Issues permission!**

For all existing synchronizers, the creator's username can be found in the Run as User column on the synchronization settings page. Before executing the transaction, Jira validates the user's permissions and then records the result together with the username in the log.

Changing Permission to Manage Synchronizers

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

By default, any logged-in user with Control access level(see page 543) for a structure can manage that structure's Synchronizers(see page 983). However, you can restrict this ability to one or more user groups.

To select who can manage synchronizers:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Select Who Can Control Synchronizers**.
3. Select whether synchronizers can be managed by **Anyone with control access to the structure** or by **Users in selected groups/roles**.
4. In the latter case, change the **selected groups/roles** list by selecting the second radio button and use the **Add Group/Role** section to add one or more required user groups or project roles. To set up the required property, use the drop-down selectors to choose either the **Group** or **Project** option, then choose the required group name or project/role combination and press the **Add** button to add it to the list. If **project** is set to "Any", this means that the user should be in the specified role for any of the structure-enabled projects(see page 954).
5. You can remove a permission option by clicking the trash can icon on the right of the option.
6. Click **Apply** when done or **Cancel** to dismiss your changes.

> The user also needs Control access level for a structure to be able to manage its synchronizers.

> Users who have *Jira Administrators* global permission are always allowed to manage synchronizers.

9.16.5.9 Protection from Synchronizer Cycles

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

It is possible to accidentally create a pair of synchronizers that would contradict each other. For example, a sub-tasks synchronizer(see page 993) can be configured to put a sub-task under an issue, while a links synchronizer(see page 997) with the "links primacy" option would have to move it to the top level of the structure.

If both such synchronizers are enabled (i.e. perform automatic synchronization), they will end up in an endless cycle, processing and overriding one another's changes, forever. These situations are undesirable, because they put unnecessary load on the server and quickly fill up issue and structure histories with meaningless change records.

Structure is designed to detect and stop such infinite cycles. In the background, Structure keeps track of how many times each of the enabled synchronizers has been invoked to process the changes generated by another synchronizer. If this number passes a certain threshold (10 by default), and there were no user-generated changes between the invocations, Structure will flag this as a probable conflict, and perform one or more of the following actions, depending on the configuration:

- Prevent one of the synchronizers from running this particular time, but keep both synchronizers enabled.
- Disable both of the synchronizers involved in the cycle.
- Send e-mail notifications to the user (or users) who created the conflicting synchronizers.
- If the synchronizers are not automatically disabled and keep cycling, send e-mail notification to the Jira administrators (all users having the "System Administrator" global permission).

The default behavior is to disable the conflicting synchronizers and send e-mail notifications to the users who installed them.

How do I respond to a cycle warning?

If you've installed a Structure synchronizer and then receive a cycle warning e-mail from Structure, please take appropriate measures to reconcile the synchronizers – disable or reconfigure one or both of them. If the second synchronizer was created by a different user, you may need to cooperate with them to solve the problem. If you're not sure what to do, contact your Jira administrators or the ALM Works support team.

Jira administrators can configure the cycle guard as described in the Advanced Configurations Guide(see page 0).

9.16.5.10 Bundled Synchronizers

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

There are several bundled synchronizers included with Structure. Other synchronizers can be provided by other Jira add-ons.

Synchronizers are a now a dark feature, meaning they are hidden from users by default. To enable synchronizers, use the Structure Dark Features and Fine Tuning Interface(see page 974).

Sub-Tasks Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

The Sub-Tasks Synchronizer automatically places sub-tasks under their respective parent issues in the structure.

This synchronizer is available only when Sub-Tasks are enabled in your Jira instance and you have at least one Sub-task issue type defined.

Sub-Tasks Synchronizer Parameters

You can select which sub-task issue types the synchronizer works with. Issues of other issue types will not be affected.

This synchronizer only supports Import / Resync into Structure (more about resync(see page 989)).

Sub-Tasks Synchronizer Rules

- When there is a sub-task (of one of the selected types) and its parent issue is in the structure, the sub-task is also added to the structure and placed under its parent task.
- The parent issue must be in the structure already - the synchronizer does not add the parent AND sub-task, nor does it add a parent if the sub-tasks is already added.

  You can add parent issues to structure manually, or use a Saved Filter synchronization to add parent issues (and probably sub-tasks) automatically.
- If a sub-task is already in the structure, and is located under a different parent (or at the top level), it will be moved under its *subtask parent* (with all sub-issues that it may have).
- Changes in structure are not synced back to sub-tasks: if you place an issue under another issue, it will not become a sub-task.
  - If you move a sub-task away from its parent task, it will soon be moved back by the synchronizer.

Filter Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

The Filter Synchronizer automatically adds issues to a structure or removes issues from a structure based on a Saved Filter or a JQL query.This powerful synchronizer lets you control the contents of the structure with an issue

filter (either a **Saved Filter** or an arbitrary **JQL Query**). It can be configured to automatically add or remove issues from a structure, or both.

Filter Synchronizer Parameters

Filter	This is where you determine the Saved Filter or JQL Query to sync with. Click **Select** to choose a saved filter or switch to **JQL query** and enter the JQL. ⚠ When this synchronizer is enabled and runs in background, it "listens" to Jira events about issues being changed. If the result of a query change without an issue being changed, the synchronizer will not detect the change and will not update the structure - for example, if you use JQL query `updatedDate > startOfMonth()`, the synchronizer will not update the structure at the beginning of a month, when the result of the query changes. You will need to do a Resync(see page 989) or use scheduled synchronization.
Add	Turns on **Add Mode**: the synchronizer will make sure that all issues from the filter's result are present in the structure.
Place added issue at the top level	If this is selected, the newly discovered issues from the filter result are placed at the top level, at the end of the structure.
Place added issue as a sub-issue of ...	Allows you to selecte an issue key (like PROJECT-123) of an issue that will serve as the parent for any newly discovered issues from the filter. These new issues will be placed as children of the selected issue, at the end of the current children list. ⚠ If the specified issue is not present in the structure, newly discovered issues will not be added to the structure.
Allow move	This option is only available if you select the "Place added issue as a sub-issue of..." option. This tells the synchronizer what to do if a matching issue is already added to the structure but located under a different parent issue. If the option is on, the synchronizer will move the issue (with all its possible sub-issues) under the specified parent issue. If the option is off, the issue will be left alone where it now resides.

Remove	Turns on **Remove Mode**: the synchronizer will remove issues from the structure when they no longer are present in the filter result. However, if an issue to be removed contains sub-issues that should stay in the structure, it will not be removed. **CAREFUL!** If Remove mode is enabled and another synchronizer is adding issues (for example, a Sub-tasks synchronizer), it could result in an endless cycle. Structure has safeguards in place to protect against this, but it is best practice to avoid this situation whenever possible. See Protection from Synchronizer Cycles(see page 991) for more details.
Remove only from where added issues are placed	Additional flag to remove issues only if they are either at the top level or under the issue where they were initially placed by the synchronizer. If this option is enabled, and you move an automatically added issue somewhere else, it will not be removed, even if it is no longer present in the search result.

This synchronizer only supports Import / Resync into Structure (more about resync(see page 989)).

If the Saved Filter used in the configuration is later deleted, or if you lose permissions to run it, the synchronizer will not work.

No matter how synchronizers are configured, they will only affect issues from the projects that are enabled for synchronization(see page 954).

Filter Synchronizer Rules

- The synchronizer adds issues from its filter's result to structure and/or removes issues that no longer are in the filter's result.
- Whenever an issue changes, a query is run to see if it matches the filter. On resync, all issues are checked.
- With **Add** mode on, an issue will be added to the structure if it matches the filter - even if the user has manually removed it from there. If the issue is already in the structure, it will not be affected, unless **Allow Move** is on - in which case it will be relocated under the specified parent issue.
- With **Remove** mode on, an issue will be removed from the structure if it does not match the filter - even if a user manually added it to the structure.
- When adding issues on Resync or Import, the synchronizer places them at the end of the structure (or under the specified parent issue), in the order that corresponds to the filter's order. However, if only part of the filter result is added (for example, because other issues are already in the structure), the final sequence of issues may be different from the filter result.

Automatic Branch Removal

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

The “Double-check sub-issues” option is useful when you want to build and, more importantly, maintain a structure where you have:

- A certain set of issues on the top level
- All issues linked to those issues placed beneath them

Specifically, the Double-Check option is necessary for removing top level issues (and all the issues linked to them), when the top level issue no longer passes the synchronizer filter.

Example
You are trying to build a structure with all Open Stories on the top level and the issues that block them added below.

To build this structure you will need to configure the Filter synchroniser(see page 993), which will add Open Stories to the top level, and the Links synchroniser(see page 997), which will add linked issues.

To get the list of the top level issues, you could use the following JQL query:

```
issuetype = Story and status = open
```

However, if you use this as the filter query and select the remove option, the Filter Synchroniser will remove all children added by the Link Synchronizer, because the children do not pass the JQL query.

To solve this problem, you can extend the query with an S-JQL expression(see page 428), which returns both parents and their children, which are already in the structure - this will prevent the Filter Synchronizer from removing children from the Structure:

```
(issuetype = Story and status = Open) or issue in structure(“Open Stories Structure“,
"issueOrAncestor in [type = Story and status = Open]”)
```

Now the last step is to automatically remove any stories (and issues linked to them), when the status of the story changes. If the **Double-Check** option is not selected, once the Story status changes, the synchronizer will see that the Story should be removed, but will think that the children still pass the filter (because there were no explicit changes done to them). As a result, it will keep both the Story and the children in the Structure. Selecting the **Double-Check** option will force it to check if the children still pass the filter - and it will remove the whole branch.

Links Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

The Links Synchronizer maintains issue links between parent issues and child issues.You can use this synchronizer to replicate the hierarchy in the structure with issue links, or to import a hierarchy that was previously created with links.

The Links synchronizer is available only when Links are enabled and there is at least one link type.

This synchronizer supports Resync in both directions (Import and Export) (more about resync(see page 989)). Incremental synchronization watches both structure changes and issue link changes and applies those change to the other side (unless the Reverse contradicting changes(see page 998) option is specified).

No matter how synchronizers are configured, they will only affect issues from the projects that are enabled for synchronization(see page 954).

Synchronizers can add and remove Jira issue links based on the permissions of the user that installed the synchronizer.

Links Synchronizer Parameters

Link Type
The type of link to sync with. Links of other types will be ignored.

Link Direction
Defines which side of the link is the parent issue and which is the sub-issue.

Parent Issue Filter and Sub-Issue Filter
If set, these filters determine which issues and links can be affected by the synchronizer:

- If a link's parent issue or sub-issue (as determined by **Link Direction**) doesn't pass the corresponding filter, then the link is ignored by the synchronizer. For example, if there are two issues that belong to the structure and pass the corresponding filters, but one of them falls out of its corresponding filter, the link will not be deleted.
- If there is a parent issue and a sub-issue in the structure, and either of them doesn't pass the corresponding filter, the synchronizer will not create a link between them.

You can use saved filters or JQL queries.

Scope
Defines which issues are affected by the synchronizer, based on whether they are in the structure or not.

- **Synchronize issues that are already in the structure** means that the synchronizer will affect only those issues that are already in the structure or reachable from it via issue links. Use this option when you need manual control over which of the linked issues appear in the structure.
    - If **Expand to sub-issues** is selected, the synchronizer will add sub-issues to the structure if their parent issue is in the structure.
    - If **Expand to parent issues** is selected, the synchronizer will add a parent issue to the structure if any of its sub-issues are in the structure.
- **Synchronize all issues that have links of selected type** means that the synchronizer will affect all issues that have matching issue links and pass the **Issue Filters**. For example, you can use this option to import all issue relationships represented by links into an empty structure.

This setting also controls which issue links can be deleted during export, manual resync *from* structure, or incremental synchronization. For example, when you remove a sub-issue from the structure, the synchronizer will remove the corresponding link only if it could have added this sub-issue back (that is, when either **Expand to sub-issues** or **Synchronize all issues** is selected).

> ⚠ **CAREFUL!** Please be careful when using this synchronizer with **Synchronize all issues** option selected, because Exporting or Resyncing *from* Structure would delete all the existing links of the selected type between issues that are not in the corresponding positions in the structure.

Removal
Defines how the synchronizer treats a sub-issue that doesn't have a link to justify its position in the structure (for example, when a user deletes the link from its parent issue):

- When **Move upwards** is selected, the synchronizer will move such an issue up the hierarchy until it's either at the top level of the structure or in a position that doesn't contradict the settings (for example, under an issue that does not pass the **Parent Issue Filter**).
- When **Remove** is selected, the synchronizer will remove such an issue from the structure, together with all its sub-issues.

Primacy
By default, when a synchronizer is installed and enabled, it tracks changes made by users and applies them to the "other side":

- When a user creates or deletes issue links, the synchronizer adjusts the structure accordingly.
- When a user changes the structure, the synchronizer creates or removes the corresponding links.

You can use the **Reverse contradicting changes** option to override this behavior and specify the primary place where issue relationships are stored:

- With **Structure primacy**, when a user creates or deletes a link that is within the scope of the synchronizer but contradicts the structure, that change will be reverted. One needs to change the structure to adjust issue relationships.
- With **Links primacy**, the synchronizer reverts changes to the structure that contradict issue links. One needs to change the links to adjust an issue's position within the structure. Note that this does not apply to reordering issues without changing their parents.

> ⚠ Primacy does not apply during Export, Import or manual Resync.

Links Synchronizer Preserves Links Between an Added List of Issues

When a list of 2 or more issues is added to the structure, links between these issues are preserved, and the issues form a hierarchy according to these links. Such a situation may arise, for example, when moving several issues into the structure at one time.

This differs from the default behavior when the **Reverse contradicting changes** option is not selected: normally, if an issue *A* is added to the structure as a sub-issue *B*, and both of them pass the Issue Filters, Links synchronizer would establish a link between *A* and *B* and remove all other links of the corresponding type where *B* is on the sub-issue end of the link. When a list of issues is added, however, the synchronizer behaves as if **Links primacy** was selected.

Links Synchronizer Rules

- When the synchronizer is enabled:
  - Changes in the structure will be reflected by creating and removing links of the selected type.
  - Links created or removed by the user will be automatically reflected in the structure.
- Links created and removed by the synchronizer are not recorded in the issue history, and the issue update time is not changed (due to performance reasons).
- Use Resync (*from* Structure to Links) or Export to update the links according to the structure.
  - If **Synchronize all issues** is selected, all other links of the selected type will be deleted.
  - Otherwise, the links that are reachable from the structure considering **Expand to...** options, but not represented in the structure, will be deleted.
- Use Resync (from Links *into* Structure) or Import to add and rearrange the issues in the structure according to the existing links.
  - If **Synchronize all issues** is selected, all issues with matching issue links will be added to the structure.
  - Otherwise, only the issues reachable from the structure considering **Expand to...** options will be added.
- Links that violate hierarchy restrictions are treated as follows:
  - If a sub issue has more than one parent issue, the most recent issue link is used.
  - If there is a sub-issue cycle, the oldest issue link is not used.
  - There is an exception to the two preceding rules: Links synchronizer prefers to use links between an added list of issues(see page 999), even if they are older than others.
  - Unused links are deleted during incremental synchronization and ignored during Import or manual Resync.

Activity Stream

By default, changed made to links by the Link Synchronizer are not written to the activity stream, because doing so may affect Jira performance.

If you want these actions to be written to the activity stream, please perform the following steps:

1. Add a new Jira startup system property: `-Dstructure.bulkLinkProcessor.useLinkManager=true` This page[639] describes how to add JIRA startup properties.
2. Restart Jira

639 https://confluence.atlassian.com/display/JIRA/Setting+Properties+and+Options+on+Startup

Agile Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

The Agile Synchronizer lets you synchronize the position of issues in the structure and on an Agile board (such as a Scrum or Kanban board) using Rank synchronization, and synchronize Epic fields with the position of stories under epics in the structure.

Agile Synchronizer Parameters

Synchronize	Choose the mode of operation: • Use Agile Board configuration • Use custom projects and fields configuration
	Agile Board mode parameters:
Agile Board	Select the Jira Agile board to synchronize with. The issues matching the Board query will be synchronized. If the structure contains other issues, they will not be affected.
Synchronize Rank	If checked, repositioning issues in the structure will also change their Rank respectively. Changing Rank in Jira will likewise cause the issue to be repositioned in the structure.
Synchronize Epics	If checked, placing an issue under an epic in the structure will add it to that epic in Jira. Similarly, if you add an issue to an epic (on the Scrum board or by changing Epic/Theme field of the issue), the issue will be repositioned under that epic in the structure.
Auto-add sub-tasks	If checked, all sub-tasks of an issue will be added to the structure as well and positioned under the parent issue.
	Custom projects and fields parameters:
Projects	Select the Jira project(s) to synchronize with. If the structure contains issues from other projects, they will not be affected. The issues from all selected projects will be synchronized using the same Global Rank field.
Rank Field	The field of type "Rank" that holds the rank (backlog order) for the selected Project. If you do not wish to synchronize rank, select *Don't synchronize*.

Synchronize Epics	If checked, placing an issue under an epic in the structure will add it to that epic in Jira. Similarly, if you add an issue to an epic (on the Scrum board or by changing Epic/Theme field of the issue), the issue will be repositioned under that epic in the structure.
Auto-add Subtasks	If checked, all sub-tasks of an issue will be added to the structure as well and positioned under the parent issue.

This synchronizer supports both Import and Export / Resync into/from Structure (more about resync(see page 989)). Incremental synchronization watches both structure changes and Jira changes and applies the change to the other side.

> ⚠ **CAREFUL!** Please be careful when using this synchronizer, especially when you add multiple issues to the Structure, as this may lead to massive updates in the Agile ranks. There is no "undo" option.

Agile Synchronizer Rules

Common Rules

- Issues that do not belong to the synchronized project(s) are not affected.
- This synchronizer does not add issues to the structure (with two exceptions, explained below). You can use a Saved Filter synchronizer(see page 993) together with an Agile synchronizer to automatically add and position issues.

Sub-Tasks Synchronization

- With the **Auto-add sub-tasks** mode on, sub-tasks are added to the structure if their parent is in the structure.
- The sub-tasks are forced to stay under their parent. If you move a sub-task somewhere else, it will jump back under the parent again. You can rearrange the order of the sub-tasks, which will be sync'ed to the Agile Rank, if the Rank Field is configured.

Rank Synchronization

- Repositioning issues in the structure causes a Rank change and the repositioning of issues on the Planning Board.
- Rearranging issues on the Planning Board causes the issues to be rearranged in the structure.
- When issues are repositioned in the structure according to Rank, they are never moved under a different parent issue.

  > ⚠ This restricts the possible rank changes in Jira - you can only move an issue to the position of another issue that is under the same parent issue in the structure; otherwise, the issue will "jump back" later.

Epic Synchronization

- Placing an issue under an epic in the structure will cause its Epic field to change to that epic.
  - It does not matter what depth the issue is at in the structure. A sub-sub-sub-issue of an epic issue will still have its Epic field updated.
- If you move an issue in the structure so that it is not under any epic, its Epic field will be cleared.
- If you manually change the Epic field to point to a different epic, the issue will be repositioned under that epic in the structure.

    - An issue that has the Epic field pointing to an epic that is in the structure will be automatically added to the structure.
- If you clear the Epic field or change it to point to an epic that is not in the structure, the issue will be moved up in the structure until it is no longer under any epic.

How to Add Issues to Structure Sync'ed with JIRA Agile
When the Agile synchronizer is enabled, it automatically updates the Agile order in background when any Structure change happens. This means that if you carelessly add issues from the sync'ed project to the structure in some random order, their ranks will be updated according to that order.

To add issues to the structure without breaking the existing backlog order:

- If adding manually on the Structure Widget, use a JQL search and add the *order by Rank* clause to the end of the query, using the rank field that is used by the synchronizer.
- Select the position of the added issues carefully (using drag-and-drop or copy/paste) - the order is likely to change unless you place issues under another issue without any other sub-issues (see *Syncing Partial Orders* below).
- If using a Saved Filter synchronizer to add issues, add the *order by Rank* clause to the Saved Filter's query. Note that any issues added by the Saved Filter synchronizer will appear at the end of the structure, and have the lowest ranking.

Syncing Partial Orders
Agile Boards are flat (except for sub-tasks), and the Structure is hierarchical - so it is not possible to precisely rearrange a structure to have all issues come in the same order as they do on the Planning Board, without changing issue parents or making the Structure also flat.

To that end, the structure syncs subsets of the issues in the hierarchy with Agile Rank. For example, consider the following Structure:

A	
	B
	C
D	
	E
	F

It is not possible to rearrange the sub-issues so they come in the following order: B, E, C, F - although this is possible on the Planning Board. Instead, the structure will synchronize sub-sets of the issues in the Structure with Jira. The following sub-sets will be synchronized separately:

- A, D - top-level issues: A must come before D on the Planning Board
- B, C - sub-issues of A are sync'ed separately, so B must come before C on the Planning Board
- E, F - sub-issues of D are sync'ed separately, so E must come before F on the Planning Board

In Jira version 6.1 and later, the epics are treated as a separate set of issues, different from Stories and other non-Epics. To accommodate this, Structure updates the rank of issues using a "partial order" approach, syncing epics and non-epics separately. This means that if an epic comes before a Story on the Structure Board, it is not required that they come in the same order on the Scrum Board.

Status Rollup Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

The Status Rollup synchronizer automatically aggregates statuses of the sub-issues and updates the status of the parent issue. For example, if all sub-issues are *Resolved,* it can update the parent issue's status to *Resolved*.

Status Rollup Synchronizer Parameters

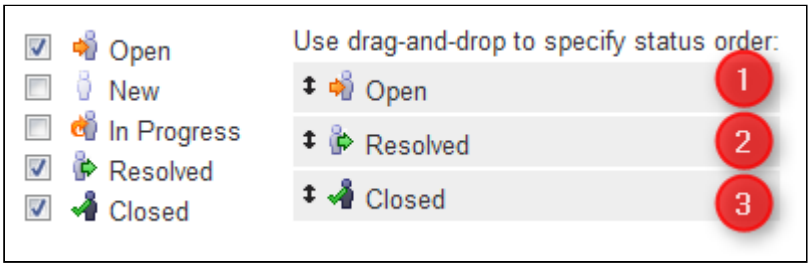
Enabled Projects	Only issues belonging to the selected projects are **changed**. It does not matter what project sub-issues belong to, as long as their parent belongs to the enabled project — every sub-issue counts with its status.
Enabled Issue Types	Determines which types of issues can be changed by the synchronizer. As with the enabled projects, only the parent issue type is checked.
Statuses Rolled Up	The selection and order of statuses that are used to calculate the parent issue status. Parent issue status is set to the *earliest* status among its sub-issues. If a sub-issue has a status not selected in this parameter, the parent issue is not changed.
Allowed Transitions	For every status, you can select which transitions the synchronizer can make to move an issue to that status.
Resolution	The value the *Resolution* field is set to when a workflow transition requires it. By default, a current or default value for Resolution is used.

The synchronizer is normally installed, resynced and used in the Incremental mode, tracking changes to issues and the structure and updating issues based on those changes. The synchronizer supports Exporting from Structure, changing statuses of the issues in the structure on a one-time basis.

How the Status Rollup Synchronizer Works

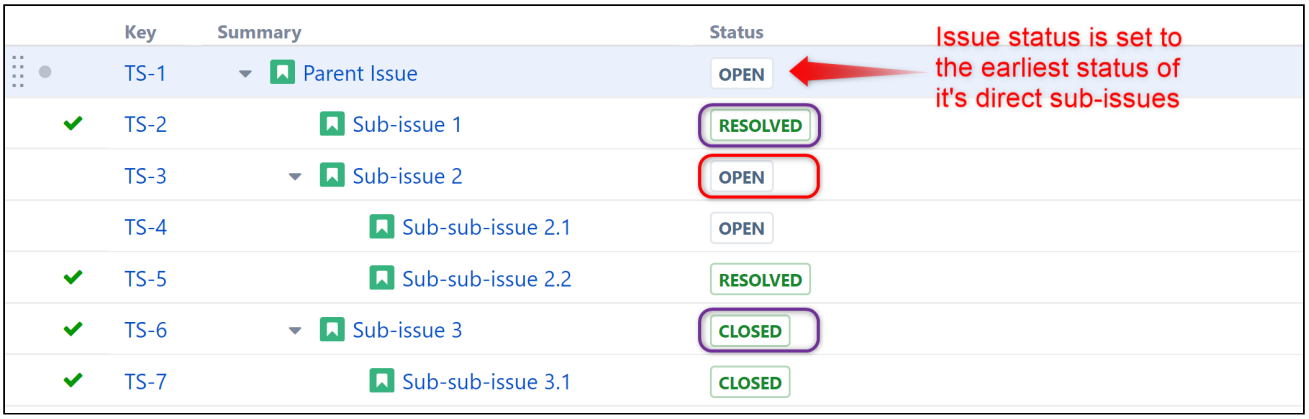
The synchronizer tracks updates to issues and to the structure, and then tries to make sure that the status of the parent issue corresponds to the aggregate status of its direct children.

When you configure Status Rollup, the most important parameter is the selected Statuses and their order:



Statuses that are not selected in the parameters are not recognized by the synchronizer. If a sub-issue has one of the unselected statuses, the synchronizer does not change the parent issue.

The order of the selected statuses should correspond to *earliest-to-latest* order of phases of the workflow. For example, the screenshot above shows a configuration where *Open* is followed by *Resolved*, which is followed by *Closed*. With that configuration, once all sub-issues of an issue are *Resolved*, the synchronizer will try to make the issue *Resolved* too. Once all sub-issues are *Closed*, the issue will be made *Closed*. But if at least one sub-issue happens to be *Open*, the issue status will be set to *Open* — because it is the earliest status in the specified order.



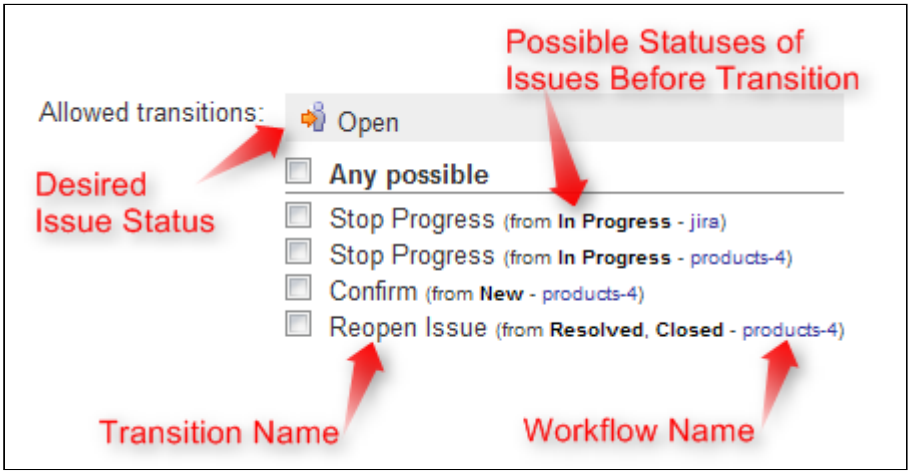
On the screenshot above:

- **Sub-issue 3** has a single sub-issue, which has status **Closed** — since all of its sub-issues are closed, Sub-issue 3 should be **Closed** too.
- **Sub-issue 2** has one **Open** sub-issue and one **Resolved** sub-issue — it should be **Open**, because Open status comes before Resolved in the order specified earlier.
- **Parent Issue** has sub-issues that have statuses **Open**, **Resolved** and **Closed** — so it should be **Open** for the same reason. Once Sub-issue 2 is **Resolved**, Parent Issue will be automatically **Resolved** (because that will be the new earliest status). Once all sub-issues are **Closed**, Parent Issue will automatically be **Closed**.
- **Sub-issue 1**, **Sub-sub-issue 2.1** and **Sub-sub-issue 2.2** do not have sub-issues of their own, so the synchronizer does not change their status.

If one of the sub-issues gets a status not listed in the synchronizer configuration, the synchronizer just skips the issue. For example, if we change the status of **Sub-issue 2** above to **In Progress**, **Parent Issue** will not be updated. If we then change the status of **Sub-issue 2** to **Resolved**, **Parent Issue** status will be updated to **Resolved**.

How Status is Changed

Jira allows a status to be changed only through a workflow transition, so the only way the Status Rollup synchronizer can set the desired status on an issue is to apply a workflow transition. Therefore, when you select a status, you also need to select which transitions the synchronizer is allowed to make.



The synchronizer:

1. Checks what status the issue currently has;
2. Calculates what status it should have, based on the statuses of sub-issues;
3. Finds workflow transitions that can transfer the issue from the current status to the required status;
4. Checks which of those transitions are allowed by the configuration; and
5. Tries to apply matching transition number one. If that fails, it tries the next, and so on.

Note that all transitions are done under the account of the user who installed the synchronizer.

Why Can a Workflow Transition Fail

It's not guaranteed that the synchronizer will be able to change the Status, because workflows are too flexible, and there are many reasons that a given transition which you have allowed in the configuration can fail to execute. Here are some of the possible causes:

- You (the user who has installed the synchronizer) do not have the required permissions to make the transition
- You are not the Assignee of the issue — required for In Progress status
- Some other pre-condition defined in the workflow fails
- The workflow transition requires a field to be set on an issue that has no default value

As described above, it's possible that there are several possible transitions from one status to another. The synchronizer will try all of them unless one of them succeeds.

If the synchronizer fails to update the status, a warning message will be written into the server logs (subject to logging configuration).

Changing Resolution

You can set up a specific *Resolution* value to be set whenever a transition involves changing the resolution. If you don't specify this parameter, the default resolution or already existing resolution will be used.

In order to tell which issues have been automatically moved to a status like Resolved or Closed, you can set up a special resolution such as *Auto-Resolved*.

Manually Changing Status of an Issue That Has Sub-Issues

Even if an issue has sub-issues and is subject to Status Rollup, you can manually change its status. Although the synchronizer will **not** be forced to recalculate the status of that issue immediately, it will recalculate the status if any of the sub-issues change – probably reversing your change - if it finds an allowed transition.

If you'd like the synchronizer to only move issues *forward* (for example, from *Open* to *Resolved*), but not vice versa, you can configure the allowed transitions accordingly.

9.16.5.11 Undo Synchronizer Actions

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

Caution should always be exercised when using synchronizers. An incorrectly configured synchronizer or an accidental move can result in unexpected changes to both a structure and Jira.

Fortunately, Structure provides a method for undoing changes made by synchronizers.

Synchronizer Audit Log

To review actions that were completed automatically by installed synchronizers:

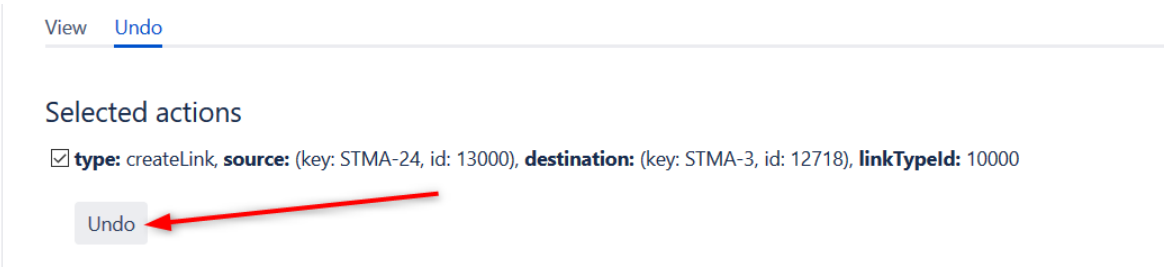
1. Go to the Administration menu and select **Structure | Support**.
2. Under Structure Support, locate the Synchronizer Audit Log section and click **View Synchronizer Audit Log**.
3. On the Synchronizer Audit Log screen, scroll down to view a list of recent synchronizer actions. If you do not see the action(s) you wish to undo, you can search the audit log based on timeframe, Sync Instance ID and Structure ID.

If you do not have access to the Administration menu, speak with your Jira administrator.

Undo Synchronizer Actions

Once you have identified the action(s) you wish to undo, select them within the Audit Log and click **Undo**.

On the next screen, review and confirm your selection by clicking **Undo**.



9.16.5.12 Copying Synchronizers

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 140)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

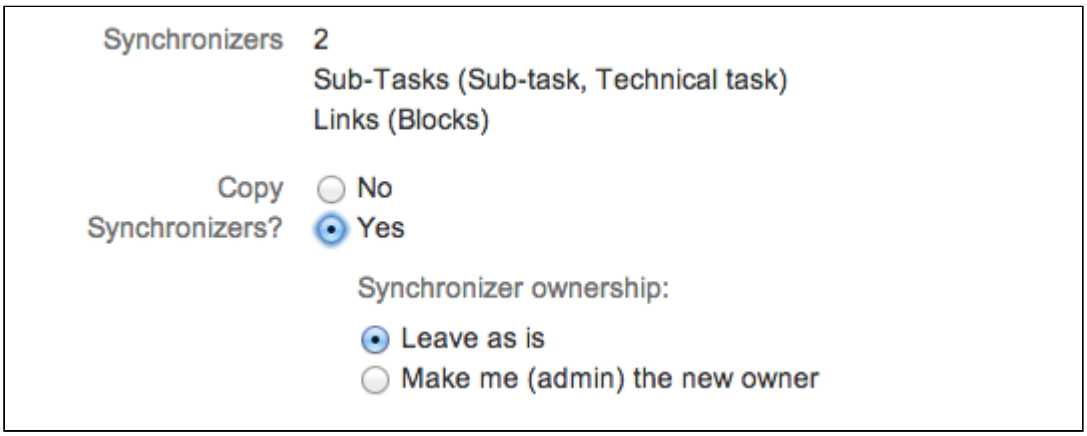
To enable synchronizers, see Advanced Configuration and Dark Features(see page 974).

When copying a structure[640] that has synchronizers, you can use the **Copy Synchronizers** option to make Structure create a copy of every synchronizer installed in the original structure.

If you don't see the **Copy Synchronizers** option, you may not have permission to create synchronizers.

640 https://wiki.almworks.com/display/structuremaster/Copying+a+Structure

Copying Parameters



You can decide to leave the original ownership of a synchronizer ("**Run As**" parameter) or make yourself a new owner for each of the copied synchronizers.

⚠ Only Jira administrators can change the synchronizer ownership.

⚠ Making yourself the new owner means that all synchronizers in the copied structure will run under your account.

Required Permissions

To be able to copy synchronizers, you need permissions to create and configure synchronizers(see page 991).

Enabling Copied Synchronizers

After the structure is copied, all copied synchronizers are initially disabled. To enable them:

1. Review their configuration to ensure they still meet your needs
2. Adjust as necessary
3. Run **Resync & Enable** - this does an initial sync and enables the synchronizers to run in the background

9.17 System Requirements

9.17.1 Atlassian Platform

Jira Versions Supported	8.20+ (by the latest version) See also: Platforms supported by Jira[641]

641 https://confluence.atlassian.com/adminjiraserver071/supported-platforms-802592168.html

Jira Editions Supported	Jira Core, Jira Software, Jira Service Management (formerly Jira Service Desk)
Jira Data Center	Supported See Server Requirements below
Confluence Versions (Structure.Pages)	7.4 – 8.3 + (by the latest version)

9.17.2 Databases

Databases used by Jira are also supported by Structure.

9.17.3 Browsers

Structure Plugin is compatible with the following browsers:

Browser	Supported versions	Versions known to NOT work
Mozilla Firefox	All recent versions	
Chrome	All recent versions	
Safari	All recent versions on OS X	Safari for Windows is not supported.
Edge	All recent versions	Internet Explorer is not supported
Other browsers	Unsupported, but may work	

9.17.4 Server Requirements

- At least 100MB of free disk space is needed on the server. See Structure Files Location[642] for details.
    - On Jira Data Center, each node must have sufficient free disk space in the local home.
- Java process running Jira needs at least an additional 200 MB of heap memory. If running on Java 7 or earlier, ensuring sufficient free PermGen space is recommended. See Memory Guidelines(see page 944) for details.
- Jira process must have read/write permissions to the Jira (local) home directory to create the `structure` sub-directory automatically.

642 https://wiki.almworks.com/display/structure/.Structure+Files+Location+v8.3

9.17.5 Non-Conforming systems

With regards to systems that don't conform to Jira requirements and Structure requirements: while we sometimes know that a specific configuration doesn't work, more often it's a grey area so feel free to try and let us know the results.

9.18 Confluence Gadget - Admin Configuration

The following articles will assist admins with enabling and configuring the Structure gadget within Confluence.

9.18.1 Adding Structure Gadget to Confluence Configuration

To enable Structure Gadget in Confluence, you first need to create an application link and configure that link.

The following Atlassian documentation will help you get started.

1. Unless you'd like to see Structure as an anonymous user, connect Confluence to Jira using **Application Links**. You'll need to enable outgoing authentication from Confluence to your Jira server.
   Documentation: Configuring Application Links[643]
    a. Use **OAuth Authentication** to let the Confluence page viewer authenticate separately with Jira. (Preferred)
       Documentation: Configuring OAuth for an Application Link[644]
    b. Use **Trusted Applications** authentication if you'd like Confluence users to act in Jira under the same usernames without additional authentication.
       Documentation: Trusted Application Authentication[645]

       > Structure Gadget can be enabled to allow modifications to the structure, updating and creating issues under the account that is used by Confluence to access Jira. Make sure you understand how Trusted Applications work before allowing production structures to be accessed with this kind of authentication. Using OAuth is more secure, because the end-user will never be able to do anything that they are not able to do directly in Jira.

2. Add Structure Gadget to the list of **External Gadgets**. Remember that you can copy the URL of the Gadget from the gadgets selection dialog, when you click **Add Gadget** on the Jira dashboard.
   Documentation: External Gadgets[646]
3. Check the sample page to confirm whether you can include the Structure macro and get data from Jira.

643 https://confluence.atlassian.com/doc/linking-to-another-application-360677690.html
644 http://confluence.atlassian.com/display/APPLINKS/Configuring+OAuth+Authentication+for+an+Application+Link
645 http://confluence.atlassian.com/display/DOC/Configuring+Trusted+Applications+Authentication+for+an+Application+Link
646 https://confluence.atlassian.com/display/DOC/Registering+External+Gadgets

9.18.1.1 Troubleshooting

If you have problems using the Structure gadget in Confluence, check the browser's console. If you see errors saying that loading some of the resources is denied, it is likely due to a CORS problem in Jira. To work around that problem, see Setting Up CORS Filter in JIRA(see page 1011).

9.18.2 Setting Up CORS Filter in JIRA

Sometimes Structure Gadget fails to load correctly in Confluence. You might see missing icons or the application can fail to work.

This may happen because of a known Jira issue that prevents the Structure gadget from loading resources from Jira when it's being served in Confluence on another web domain.

To work around this problem, you can set up a CORS filter in the Tomcat server that runs Jira (Nginx users may want to consider this alternative Nginx Configuration Option(see page 1011)):

1. Copy cors-filter-2.4.jar, java-property-utils-1.9.1.jar from CORS docs[647] to the **/lib** directory under Jira's installation folder.
2. Edit the file **JIRA_INSTALL_DIR/atlassian-jira/WEB-INF/web.xml** and add the following:

```
 <!-- ==================== CORS configuration ====================== -->
<filter>
    <filter-name>CORS</filter-name>
    <filter-class>com.thetransactioncompany.cors.CORSFilter</filter-class>
    <init-param>
        <param-name>cors.allowOrigin</param-name>
        <param-value>http://YOUR-CONFLUENCE-DOMAIN.com</param-value>  <!-- use
http: or https: depending on your configuration -->
    </init-param>
    <init-param>
        <param-name>cors.supportedMethods</param-name>
        <param-value>GET, POST, HEAD, OPTIONS, PUT, DELETE</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>CORS</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

3. Restart Jira

9.18.2.1 Nginx Configuration Option

Some Nginx proxy users reported that adding the following block directive is an effective workaround for addressing the missing CORS headers issue.

Please replace ***XXXX*** with your actual Confluence domain name:

647 http://software.dzhuvinov.com/cors-filter-installation.html

```
location ~* \.(eot|ttf|woff|woff2)$ {
    add_header Access-Control-Allow-Origin "https://confluence.XXXX.com";
    try_files $uri @jira;
  }
  location / {
    try_files $uri @jira;
  }

  location @jira {
    proxy_pass http://localhost:8080 ;
    proxy_read_timeout 180s;
    proxy_http_version 1.1;
    proxy_redirect off;
  }
```

9.19 Clearing Structure Caches

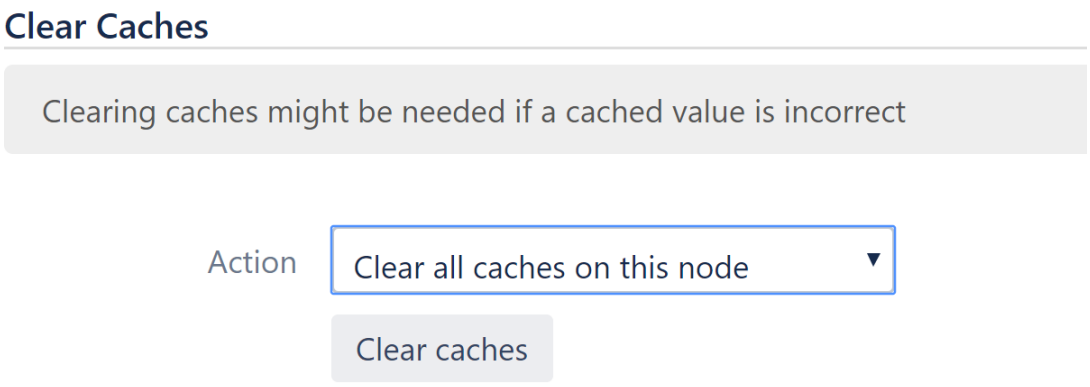
Structure caches can be cleared for specific users or the whole Jira instance. In Jira Data Center, it's possible to clear caches for the current node or all nodes in the cluster.

⚠ Jira's own caches are not affected by this action, only Structure caches are cleared.

There are two ways to clear Structure caches:

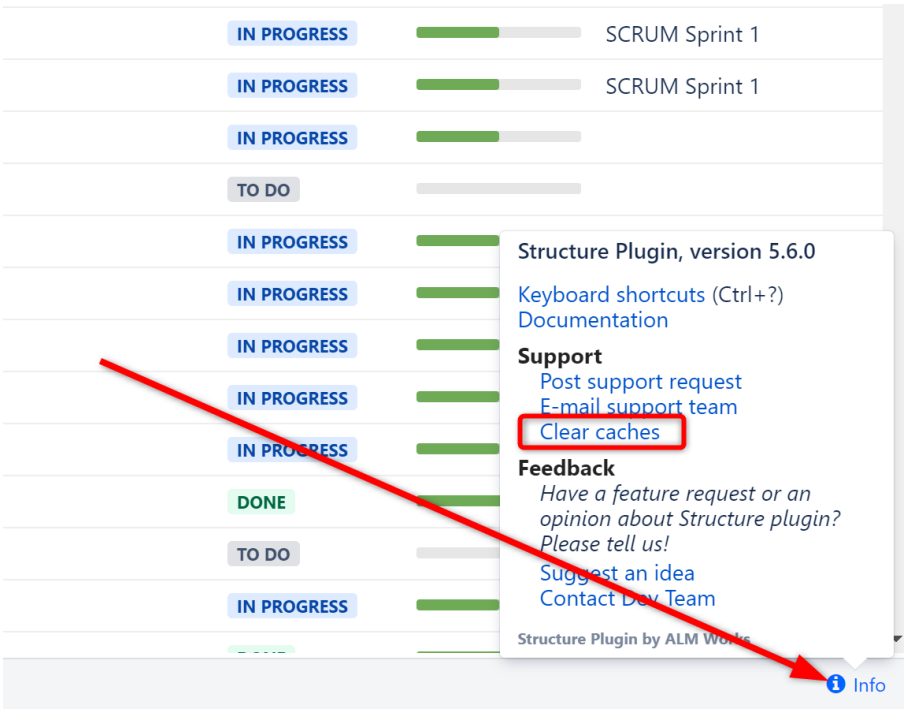
Admin-only Method

1. Navigate to **Administration | Structure | Support**
2. Scroll down to the **Clear Caches** section and select the caches to be cleared.
3. Click **Clear Caches**.



All-user Method

The Clear caches option is also available to admins and Structure users via the Info pop-up in the bottom right corner of the Structure panel.

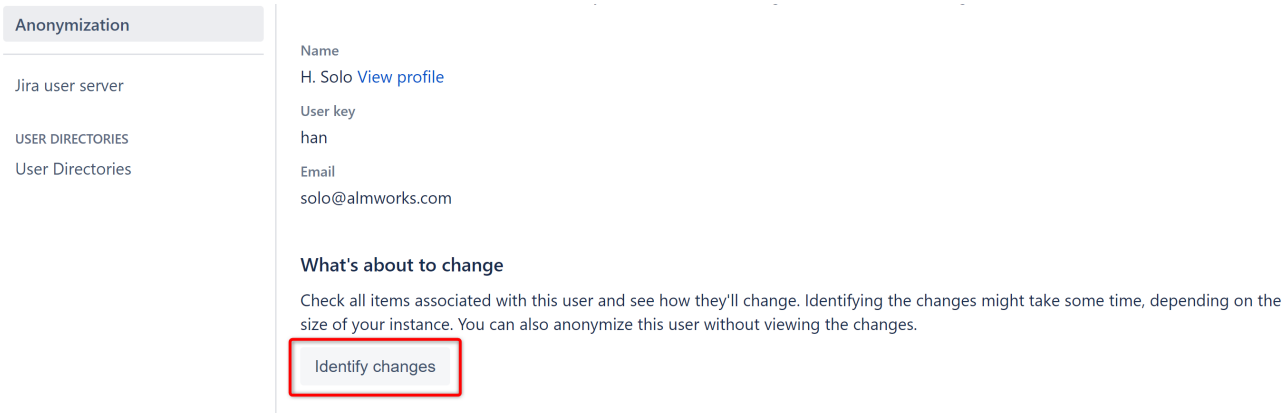


9.20 Anonymizing Users

When you Anonymize a user[648] in Jira (8.7+), any private structures, views or Effector processes owned by the user will be deleted. Shared structures or views owned by the user will be transferred to another user. Statistics related to the user will be anonymized.

9.20.1 Review User's Structures

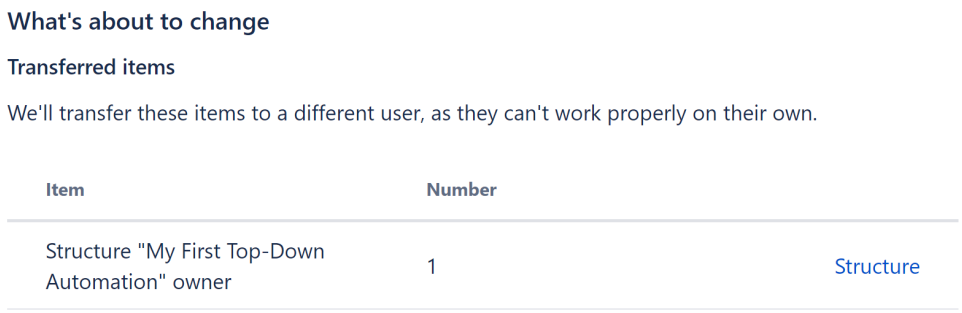
To review the structures and/or views owned by the user, on the Anonymization screen, click **Identify changes**.



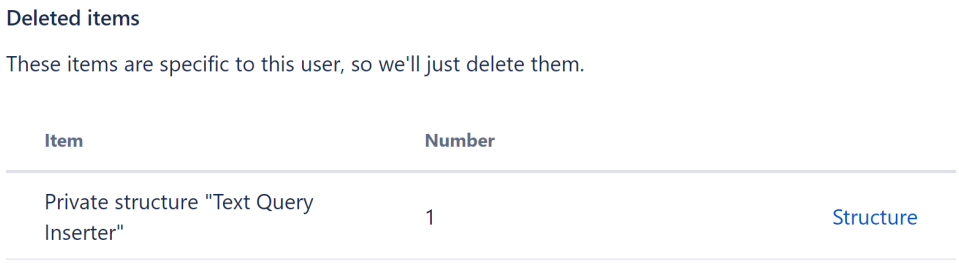
Any shared structures or views will appear under the **Transferred items** list. These will be transferred to a new user after the anonymization.

---

648 https://confluence.atlassian.com/adminjiraserver/anonymizing-users-992677655.html



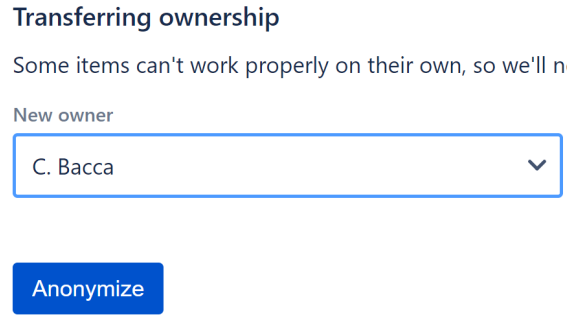
Any private structures or views (not visible to anyone but the user) will appear under the **Deleted items** list. These will be deleted.



To review any of the structures on these lists, click the **Structure** link to the right.

9.20.2 Transfer Ownership

Ownership for any shared structures or views will be transferred to the user you select in the **Transferring ownership** section.



9.21 Structure Troubleshooting

- Troubleshooting Performance and Stability Issues(see page 1025)

9.21.1 Collecting Support Zip

ALM Works support may ask you to collect a Support Zip during a support case investigation.

> To collect Support Zip, you will need **System Administrator** permissions in your Jira. You will also need a way to transfer files from the host that runs Jira instance.
>
> If you do not have the required access, please ask your Jira administrator or your system administrator for assistance.

To collect a Support Zip please do the following:

> Please note that the instructions below apply to Jira versions 7.4 and higher. In older Jira versions the 'Support tools' section has slightly different naming of options.

1. Open **Administration | System | Logging and Profiling** page.
    a. Enter STRUCTURE TROUBLESHOOTING into the **Optional Message** field, turn on **Log Rollover** and press **Mark**.
    b. Scroll down and click **Configure logging level for another package**, enter package name `com.almworks` then select logging level DEBUG and click **Add**.
2. Reproduce the problem being investigated.
3. Open **Administration | System | Troubleshooting and support tools**, switch to **Create support zip** tab. **Customize zip** and select options: Select options **Jira configuration**, **Health checks**, **Jira application logs**, **Tomcat logs**, **Thread dumps**. Unselect all other options. **Save** and **Create zip**. Once it is created, **Download zip**.
4. Send the resulting ZIP file using the upload link, provided in the support request.
5. After you've collected the support zip, you can go back to **Administration | System | Logging and Profiling** page and set the logging level for *com.almworks* to WARN - it's the default level.

9.21.2 HAR Network Report

HAR Network Report is something we (ALM Works Support) may ask you to collect, to help us understand a tricky problem that we could not reproduce.

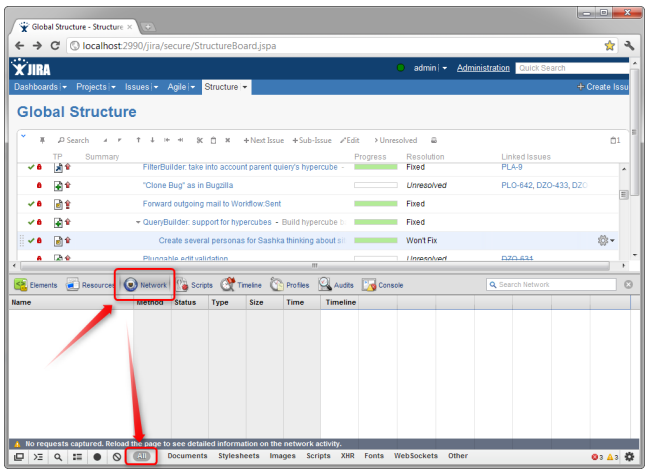
> HAR stands for HTTP Archive Format[649], a text-based format for the log of network communications between a user agent (the browser) and a web server. You can also use this report with a HAR Viewer[650] for in-depth analysis of your JIRA page load performance. (Be aware though that with an online viewer you may transfer sensitive or security-related information to a third party.)

9.21.2.1 Collecting HAR Report with Google Chrome

1. Open a new Chrome window and navigate to the page where the problem happens.

649 http://www.softwareishard.com/blog/har-12-spec/
650 http://www.softwareishard.com/blog/har-viewer/

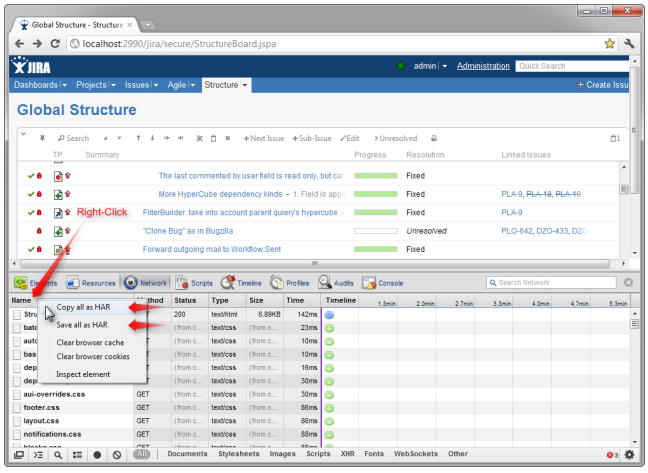
2. Press **Ctrl+Shift+I** or use menu **Wrench | Tools | Developer Tools** to open a section with developer tools. Switch to the **Network** tab there. Make sure **All** tab is selected below.



3. Reload the page by using **Ctrl+R** or clicking the Reload button. This will make Network tab log all network exchange during page load.

> The network tab will start collecting information or network exchange automatically after it's opened. If you know that the problem is not related to the initial page load, you may skip this step to avoid adding extra data to the log. If unsure, reload the page to collect the full report.

4. Reproduce the problem being analyzed.
5. After the problem has been reproduced, **right-click** on the **Name** column in the Network tab and choose either **Save all as HAR** or **Copy all as HAR**



6. Paste the report into an e-mail to our support, or attach the saved .HAR file.

9.21.3 Troubleshooting Synchronizers

Structure synchronizers(see page 983) work in background and can lead to changes in the structures or issue data that might be hard to trace. Complex configuration rules don't make things better, so it's important for JIRA admin to be able to track which synchronizers are doing what and what has caused a particular change a user is complaining about.

9.21.3.1 Structure Audit Log

Starting with Structure 3, all standard synchronizers record all actions they have taken in the database and allow the administrator to undo the changes. Navigate to **Administration | Structure | Support | Synchronizer Audit Log** to query history or apply undo.

9.21.3.2 Log Files

To get detailed reports about what's going on, you can reconfigure your JIRA logging so that structure synchronizers can produce more verbose messages. Also, you might want to direct messages from the synchronizers into separate log files.

The appearance of detailed synchronizer messages is governed by the log level: the lower the log level, the more detailed messages can appear. By default, log level for structure synchronizers is `WARN`, and you can set it to lower levels, like `DEBUG` (the lowest one.) You can set the logging level either temporarily(see page 1017) (until the next JIRA restart) or permanently(see page 1017).

To see the list of possible log levels and other general information regarding logging in JIRA, please refer to JIRA logging documentation[651].

Temporarily change log level for structure synchronizers

If you set log level in this way, it will not persist after you restart JIRA. This is a relatively simpler way than setting the log level permanently.

1. Log in as a user with the JIRA System Administrators[652] global permission.
2. Select *Administration* | *System* | *Troubleshooting and Support* | *Logging & Profiling* (tab). The 'Logging' page will be displayed, which lists all defined log4j categories (as package names) and their current logging levels.
3. Locate and click the link that reads "Configure logging level for another package", and a dialog will be displayed. For troubleshooting bundled synchronizers, specify package name **`com.almworks.jira.structure.ext`**; choose the appropriate logging level, e.g. `DEBUG`.

Permanently change log level for structure synchronizers or set up separate log files for synchronizers

This way, you need to modify the `log4j.properties` file, which is located in the JIRA installation directory[653].

The package name that all bundled synchronizers log under is `com.almworks.jira.structure`. You can add the following lines to have debug messages from synchronizers show on the console and/or in the log file (depending on their respective log levels):

```
log4j.logger.com.almworks.jira.structure = DEBUG, console, filelog
log4j.additivity.com.almworks.jira.structure = false
```

Or, you can set up a separate log file for synchronizer actions:

651 https://confluence.atlassian.com/display/JIRA/Logging+and+Profiling
652 https://confluence.atlassian.com/display/JIRA/Managing+Global+Permissions
653 https://confluence.atlassian.com/display/JIRA/JIRA+Installation+Directory

```
log4j.appender.structure-sync=com.atlassian.jira.logging.JiraHomeAppender
log4j.appender.structure-sync.File=structure-sync.log
log4j.appender.structure-sync.Threshold=TRACE
log4j.appender.structure-sync.MaxFileSize=20480KB
log4j.appender.structure-sync.MaxBackupIndex=1
log4j.appender.structure-sync.layout=org.apache.log4j.PatternLayout
log4j.appender.structure-sync.layout.ConversionPattern=%d %t %p %X{jira.username}
[%c{4}] %m%n

log4j.logger.com.almworks.jira.structure = DEBUG, structure-sync, console
log4j.additivity.com.almworks.jira.structure = false
```

9.21.4 Structured JQL Troubleshooting

If a Structured JQL(see page 428) query doesn't work as expected, please try the following steps.

1. Double-check if the query itself correctly expresses what you are searching for.
2. It's likely the Jira indexes that are used for searching have become corrupt. Please try to do a full reindex of Jira[654] — note that you should use **Lock Jira and rebuild index** option, the other one is known to not help when indexes are corrupted.
3. If the query still returns strange results, please go to the Structured JQL Troubleshooting page and follow the instructions outlined there:

   ```
   <base URL>/secure/StructuredJqlTroubleshooting.jspa
   ```

   Here, `base URL` refers to the Jira base URL[655].
   On this page, you will be able to run a Structured JQL query and collect extensive logs which we at Tempo can inspect in order to track down the issue.

Need help or have questions? Contact Tempo Support[656].

9.21.5 Collecting Performance Snapshots

Performance snapshots allow ALM Works support team to analyze performance-related problems on your JIRA server without direct access to it.

9.21.5.1 Download and install Atlas-Yourkit plugin.

Get the latest version from this page. In JIRA 4.3 and later, you can install this plugin without JIRA restart.

The performance analysis plugin and redistributed parts of YourKit profiler are free, but if you'd like to analyze the performance snapshots yourself, you'll need to obtain YourKit license and download YourKit software (they provide a free evaluation period).

654 https://confluence.atlassian.com/display/JIRA/Search+Indexing
655 https://confluence.atlassian.com/display/JIRA/Configuring+JIRA+Options
656 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

File	Modified
atlas-yourkit-0.2.jar[657]	Apr 08, 2024 by Jaramy Conners[658]

9.21.5.2 Load Profiling Agent

1. Open menu **Administration | Troubleshooting and Support | YourKit Profiling** (hint: in JIRA 4.4 and later versions, press **g,g** ("g" twice) and search for "yourkit").
2. If agent is already loaded, you'll see profiling controls - skip this step then.
3. Click **Load Agent** to load profiling agent. You'll need to have JDK installed. If you don't have JDK installed – follow the link on that page, download and install a matching JDK on JIRA host. It is not necessary to restart JIRA, just install the JDK and load agent.

> ⚠ There's certain risk that JVM will crash when loading profiling agent into JVM. A safer method of loading profiling agent is by changing JIRA start-up parameters (in `setenv.sh`/`setenv.bat`) and specifying `agentpath` parameters with other options. See YourKit Documentation[659] for details.

657 https://wiki.almworks.com/download/attachments/200605849/atlas-yourkit-0.2.jar?api=v2
658 https://wiki.almworks.com/display/~jaramy
659 http://yourkit.com/docs/10/help/agent.jsp

9.21.5.3 Capturing CPU Performance Snapshot

After profiling agent is loaded, you can click **Start CPU Sampling** on the YourKit page, then perform the actions that make JIRA slow, or wait for some time to collect the statistics. When finished, click **Stop CPU Sampling**. Performance snapshot will be saved to a directory within your JIRA Home, and the path will be shown on the YourKit page.

9.21.5.4 Capturing Memory Snapshot

Click "Take Memory Snapshot" - memory dump will be collected and saved in a file under your JIRA Home. Do not take memory snapshots unless you need to!

> ⚠ Taking memory snapshot is usually a long operation, which could last several minutes. During that time JIRA will be completely frozen. Make sure you've got enough disk space (several GBs). Don't panic - it does take that much time. After you click the button the page will be reloading. The browser may fail to load the page due to timeout - check JIRA logs to see when snapshot is finished.

9.21.5.5 Sending the Snapshots to Support Team

By default, snapshots are written into `<jira_home>/yourkit/snapshots` directory. Locate it and create a ZIP archive of all relevant snapshot files.

Please send the ZIP to us as described here: Sending Files to Support Team(see page 1024).

9.21.5.6 After Profiling Session

There's no way to unload the profiling agent. You may want to continue running JIRA with the profiling agent loaded, since it does not product much overhead. (Make sure you have stopped all the monitoring.)

For a safer / cleaner environment, you can restart JIRA. (If you made additional effort to enable profiler agent in `setenv` script, you'll need to comment that options out.)

9.21.5.7 Performance Snapshot Without Yourkit Plugin

Performance Profile allows ALM Works support team to analyze performance-related problems on your JIRA server without direct access to it.

We are using Java Profiler product called YourKit[660]. In order to collect the profile, you'll need to download freely distributed "agent" library, connect it to your JIRA instance and capture a performance snapshot. You will need to purchase a license from YourKit only if you want to analyze the captured profile yourself.

> ✓ No special knowledge is required to collect the performance profile, but being familiar with using the command-line on the server that runs JIRA helps.

660 http://yourkit.com/

Download Profiling Agent

Download the ZIP with profiling agent from here: jira-profiler-v1-yjp956.zip[661] `md5sum e3ea2b72ef4b22584c641425275050d0`

Unpack the downloaded ZIP file into the directory where you have JIRA installed (**not** JIRA home!). This will create `<jira_install>/profiler` directory under your JIRA installation path.

> ⓘ You can unpack the profiler into any other directory, but this instructions and our scripts assume that the profiler is unpacked into JIRA install dir.

If you will be able to restart JIRA before profiling, this is all you need — you can proceed to restarting JIRA with Profiling(see page 1021).

Additional Download to Profile JIRA Without Restart

If you need to profile JIRA without restarting it first (and assuming it is not already started with a profiler agent), you will need to download full distribution of the YourKit Java Profiler:

1. Open http://yourkit.com/download/index.jsp
2. Click on **ZIP Archive** type of download - **NOT** the installer! ZIP archive is typically downloaded under "Solaris" section - it is the correct link even if you run JIRA on Windows.
3. License key is not required for our purpose! Do not request evaluation license. (Unless you intend to do an evaluation of YourKit, of course.)
4. Unpack the downloaded ZIP into `<jira_install>/profiler` – this is the directory created at step 1. Unpacking will create a sub-directory there - for example, `<jira_install>/profiler/yjp-9.5.6`.

Restart JIRA with Profiling

> ⓘ If you need to profile without restart, skip this step.

> ⓘ The following instruction is provided for a standalone JIRA installation.

To restart JIRA with profiling, you need to pass additional options to Java that runs JIRA. This is done by editing `<jira_install>\bin\setenv.bat` on Windows or `<jira_install>/bin/setenv.sh` on a Unix-based OS and pointing Java to a profiler agent that you have unpacked at step 1.

1. Find out which profiler agent to use.
    a. Look into `<jira_install>/profiler/bin` directory. Typically there will be two sub-directories for your operating system: 32-bit and 64-bit. The bitness must match the bitness of JVM that runs JIRA. You can verify which Java your JIRA runs on if you open **Administration | System Info** in JIRA and look for "Java VM". If it mentions "64-Bit", then JIRA runs on a 64-bit Java.
    b. Note the name of the subdirectory under `profiler` directory that corresponds to the bitness of target JVM: it may be *win64* or *linux-x86-32* or something like that.
2. Edit `setenv` script:
    a. On Windows, set or append the following parameters to `JVM_SUPPORT_RECOMMENDED_ARGS` in `<jira_install>\bin\setenv.bat` (following is a single long line):

---

661 http://almworks.com/.files/jira-profiler-v1-yjp956.zip

```
set JVM_SUPPORT_RECOMMENDED_ARGS=-agentlib:
%~dp0..\profiler\bin\win64\yjpagent=port=10001
,onlylocal,dir=%~dp0..\profiler\snapshots,delay=20000 -XX:MaxPermSize=500m
```

b. On other OS, set or append the following parameters to `JVM_SUPPORT_RECOMMENDED_ARGS` in `<jira_install>/bin/setenv.sh` (following is a single long line):

```
JVM_SUPPORT_RECOMMENDED_ARGS="-agentpath:`dirname \"$0\"`/../profiler/bin/
linux-x86-64/libyjpagent.so=port=10001,onlylocal,dir=`dirname \"$0\"`/../
profiler/snapshots,delay=20000 -XX:MaxPermSize=500m"
```

3. Note that in the lines above, you should change **win64** or **linux-x86-64** to the name of the directory where the correct profiler agent for your OS/Java is located.
4. You may also need to change **port=10001** to make profiling agent listen on some other TCP port - in case port 10001 is already taken.
5. Stop JIRA and start it again.
6. Watch `<jira_install>/logs/catalina.out` for YourKit message like *[YourKit Java Profiler 9.5.6] Loaded.*

Use Copy & Paste to copy the parameters and then edit them in the setenv.sh

If the parameters are set incorrectly, JIRA start may fail. Verify that you have specified the agent directory correctly. Also verify that `<jira_install>` directory path does not contain spaces.

Profiler agent will use directory `<jira_install>/profiler/snapshots` to write performance snapshots - it must be write-accessible to the JIRA process.

Now you can proceed to #Running Profiling Session(see page 1023).

Attach Profiler Agent to JIRA without Restarting

If you have restarted JIRA with profiling, skip this step.

If possible, restart JIRA with profiling instead of attaching profiler agent on the fly.

You will need the full distribution of YourKit downloaded at step 1.1. You will need to run a Java program as specified below - with the same version of Java that JIRA runs on. We assume that it is in your PATH variable in the command-line, but if it's not - you need to specify a full path to `java`.

1. Find out the process ID of the process that runs JIRA. You can use `jps` command from the Java distribution.
2. Find out the location of JDK (Java Development Kit). If you don't have JDK installed (only JRE), this procedure won't work. Typically JDK home is stored in the command-line environment variable JAVA_HOME.
3. Change current directory to `<jira_install>/profiler/yjp-9.5.6`. (You may have a different version of yjp.)
4. Run the following command, substituting JIRA process ID instead of ***PID***.

a. On Windows:

```
java -cp lib\yjp.jar;%JAVA_HOME%\lib\tools.jar com.yourkit.Main -attach
PID port=10001,onlylocal,dir=<jira_install>\profiler\snapshots
```

Replace <jira_install> with the full path of the JIRA installation folder.

b. On other OS:

```
java -cp lib/yjp.jar:$JAVA_HOME/lib/tools.jar com.yourkit.Main -attach PID
port=10001,onlylocal,dir=`pwd`/../snapshots
```

The command should output something like this:

```
Attaching to process 60108 using options port=10001,onlylocal,dir=..\snapshots
 The profiler agent has attached. Waiting while it initializes...
 The agent is loaded and is listening on port 10001.
 You can connect to it from the profiler UI.
```

Running Profiling Session

To successfully run a profiling session, you need to have JIRA running with a profiling agent, as explained above. The agent does not add much overhead when being idle — it sits there waiting for your commands to start a profiling session.

General Procedure

The profiling session is controlled by sending commands to the profiling agent (within the JIRA process). The program that is used to send the commands is `yjp-controller-api-redist.jar`, located in `<jira_install>/profiler`. The common format for running this program is:

```
java -jar yjp-controller-api-redist.jar localhost 10001 <command>
```

The <*command*> is replaced with some actual command, and if you changed the default port of the agent from 10001 to something else, you need to specify that port number here instead of 10001. This command should be run from `<jira_install>/profiler` directory.

We are assuming that `java` is on your PATH. If not the case, use the full path to `java` executable.

CPU Performance Analysis

If JIRA is unresponsive or burns CPU extensively, you can run CPU analysis session.

1. Start session with the following command:

```
java -jar yjp-controller-api-redist.jar localhost 10001 start-cpu-sampling
```

2. Let JIRA work for some time. If needed, take a specific action that causes the problem to manifest.
3. Stop session and record a snapshot:

```
java -jar yjp-controller-api-redist.jar localhost 10001 capture-performance-
snapshot
```

Sending the Snapshots to Support Team

By default, snapshots are written into `<jira_install>/profiler/snapshots` directory. Locate it and create a ZIP archive of all relevant snapshot files. If the ZIP is less than 10 Megabytes, it's ok to send it to us by e-mail.

If the ZIPPed snapshot is 10 MB or larger, you need to use FTP to send it over to us:

1. Use any FTP client (`ftp` or `lftp` from the command line).
2. Connect to host `f.almworks.com`
3. Use login name `almftp` and password `almftp`
4. Upload files to the root folder.
5. After the upload is finished, please send us an e-mail with a notification that you have uploaded the snapshots.

> You will not be able to list or download files from that FTP, and your FTP client may show errors about that. That's ok and should not prevent you from uploading snapshots.

After Profiling Session

You may want to continue running JIRA with the profiling agent loaded, since it does not product much overhead. Make sure you have stopped all the monitoring.

For a safer / cleaner environment, you can restart JIRA with the profiling options in `setenv` script commented out.

9.21.6 Sending Files to Support Team

When you need to send files to Tempo support team, please use one of the following methods (listed in the order of preference).

9.21.6.1 Attach to the Support Request in Service Desk (Preferred)

File size limit: 20 MB

If the files pertain to a Support Request, please use Tempo Support[662] to upload and attach the files to the ticket. Size limit is 20 MB per upload.

662 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

9.21.6.2 Upload Files Directly to Our Server

File size limit: 5 GB

If you need to send us files larger than 20 MB, please let us know via Tempo Support[663]. We will send you a custom link that will allow you to upload such files directly to our secure server.

The files you have uploaded are safe – they cannot be accessed by anyone except Structure Support.

9.21.7 Troubleshooting Performance and Stability Issues

In cases when JIRA's performance deteriorates or if the system becomes unstable or unresponsive, it is important to achieve two goals:

1. Bring system back to normal in the shortest amount of time.
2. Collect information that would help analyze the problem and make sure it does not appear again.

The second goal is strategically very important, however, it might get overlooked in a rush to make things work "now". For example, JIRA administrator may be inclined to restart a stuck JIRA instance quickly in order for it to get back to working state as fast as possible. But if thread dumps are not collected, the developers will never know where JIRA was stuck, so the same problem may happen again.

The first goal is of course also very important. Sometimes JIRA administrator manages to restore system functioning, sometimes help from Atlassian and ALM Works support teams is needed. Support engineers and developers would typically take into account all information they have, analyze it and try to pinpoint the source of the problem. Often additional information is required from the JIRA administrator, and sending requests and replies back and forth takes precious time.

This article is intended to provide JIRA administrators with advice about how to collect maximum information about a performance or stability problem, when that problem happens. The list is not intended to be complete, additional information may still be needed, however, providing all listed information gives a good chance that a support engineer will be able to identify a problem and provide advice sooner.

- Thread Dumps(see page 1025)
- Verbose Logging(see page 1026)
- Support Zip(see page 1026)
- Browser Console Log(see page 1026)
- HAR Report(see page 1027)
- Screenshots or Video(see page 1027)

9.21.7.1 Thread Dumps

Thread dumps are the most important information when system is unresponsive or has performance issues. They allow to peek into what's going on inside JIRA's JVM process.

- Please refer to Atlassian documentation on generating a thread dump[664] for instructions of manually capturing a thread dump on the server.

663 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

664 https://confluence.atlassian.com/adminjiraserver071/generating-a-thread-dump-802593021.html

- Thread dumps are also a part of the Support Zip (3 dumps are generated in one zip), however, generating a support zip might be unavailable if JIRA is hanging.
- For best diagnosis, please collect 5-6 thread dumps with 3-4 second interval.

Please collect 5-6 thread dumps with 3-4 second interval.

9.21.7.2 Verbose Logging

If the problem has temporary but reproducible manner, you can turn on verbose logging so that the engineers can gather more information from the logs. To do so:

1. Open **Administration | System | Logging and Profiling** page.
2. Enter STRUCTURE TROUBLESHOOTING into the Optional Message field, turn on Log Rollover and press Mark.
3. Scroll down and click **Configure logging level for another package**, enter package name **com.almworks** then select logging level **DEBUG** and click **Add**.

Then you can try to reproduce the problem and collect the support zip.

Do not forget to turn off the DEBUG logging after the problem has been resolved, otherwise you may get too many messages in the logs during normal operation.

9.21.7.3 Support Zip

Support zip is the most important thing after thread dumps. It allows engineers to have full understanding of the environment and retrospect using the logs into what was going on. If you had Verbose Logging on before problem appeared, it gives even more details.

To collect a support zip:

1. Open **Administration | System | Atlassian Support Tools**, switch to **Support Zip** tab.
2. Open **Administration | System | Support Tools**, switch to **Create Support Zip** tab. Select all options. Click **Create**.
3. Download the resulting ZIP file and send it to the support teams: either attach it to the ticket, or, if the file is large, request a URL for uploading.

On JIRA Data Center, collect Support Zips on each node.

9.21.7.4 Browser Console Log

If the problem seems to be on the client side, in the browser – if there are errors or if the browser is hanging or some button or link does not respond, check out the browser's error console. Depending on the browser type, the console may be opened with different menus or keyboard shortcuts.

1. Reproduce the problem
2. Copy all contents from the console and send it to support.

Also, please include browser type and version, as well as the information about operating system.

9.21.7.5 HAR Report

HAR report is also taken on the browser and contains logs of network communications with the server. Use this log to provide information that can help troubleshoot issues with slow loading of data or general slow responsiveness on the client side.

1. Use Google Chrome
2. Open menu, More Tools | Developer Tools.
3. Switch to Network tab
4. Reproduce the problem
5. Right click in the table and select "Save as HAR with content..." or "Copy All as HAR".
6. Paste or save HAR as a file.

HAR with content provides more information but it may contain your JIRA's data. Review the contents before sending it out to support.

9.21.7.6 Screenshots or Video

When there's a visible and informative behavior demonstrated by the system, a screenshot or a video showing the problem would go a long way in getting the support engineers understand the issue.

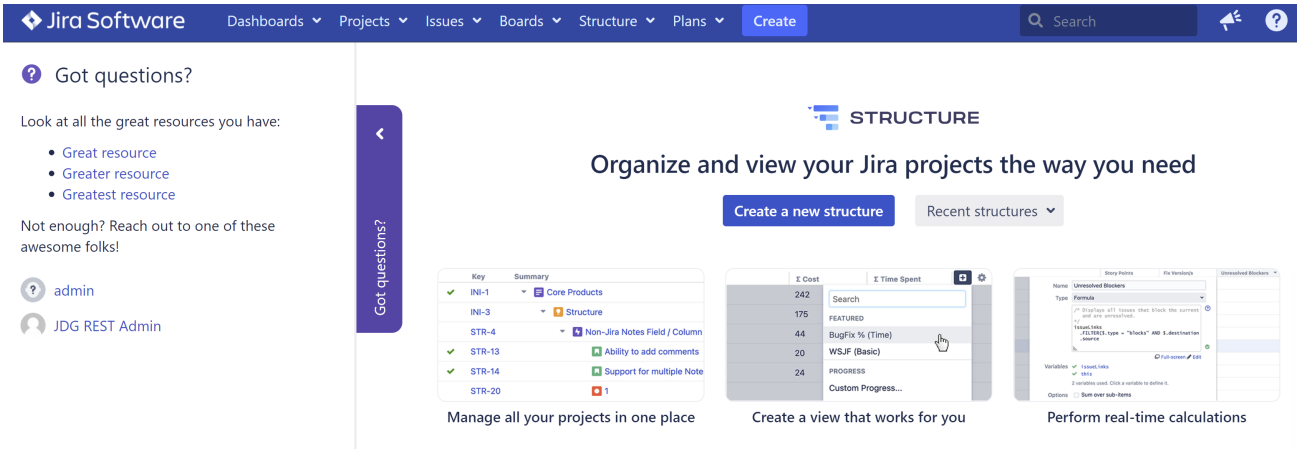
You can use operating system's native tools to capture a video, or install a third-party tool for that. Feel free to ask ALM Works Support for recommendations if you don't have preferable screen capturing tool.

See also: Monitoring and Troubleshooting Structure Usage(see page 942)

9.22 Help Tool

The Help tool allows Jira admins to create a custom help page for their teams, which can be viewed from the Welcome to Structure page.

- Give your Help tool an eye-catching title
- Add tips, links, or any other useful content
- Provide a list of support contacts within your company



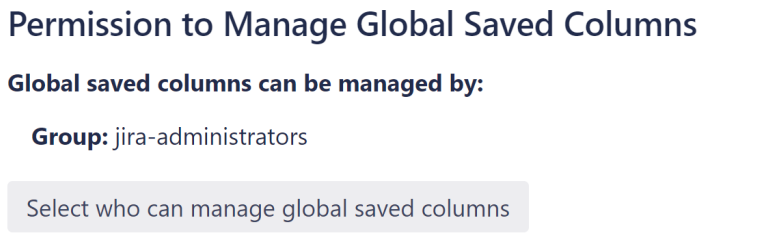
To create or edit your Help tool, go to **Administration | Structure | Help Tool**.

9.23 Managing Global Saved Columns

The Structure administrator can assign which users or groups are able to manage global saved columns(see page 524). Users with manage permission can:

- Move saved columns from the My Columns list to the Global list.
- Delete global saved columns.

To assign or remove Manage Global Saved Columns permissions, go to **Administration | Structure | Configuration**, and scroll down until you see **Permission to Manage Global Saved Columns**.



Click **Select who can manage global saved columns** to make changes.

10 Structure Developer's Guide

Structure for Developers

Structure app provides APIs that allow you to access structures, integrate your add-on with Structure and extend Structure functionality. Here are the typical use cases:

Custom Development

You customize JIRA for your customer or employer, and you need to integrate Structure with some other in-house system – see section about integrating plugins(see page 1031) and Java API reference(see page 1069).

Plugin Integration

You have your own great JIRA app, or plan to create one, and you'd like to use the issue hierarchy provided by Structure – see Accessing Structure from JIRA Plugin(see page 1031).

Extending Structure

You'd like to extend Structure, adding functionality to the plugin itself – read documentation about extending Structure functionality with additional plugins(see page 1047).

Remote Access

You need to get or change issue hierarchy remotely from some automated scripts or a client application – read about Accessing Structure Data Remotely(see page 1068) and Structure REST API(see page 1086).

10.1 Structure Developer Documentation

10.2 Structure Concepts, Developer's Perspective

This article provides an introduction to the main concepts used in Structure. Before starting your work on integration with Structure, please familiarize yourself with these concepts.

10.2.1 Basic Concepts Overview

Concept	Short Definition	API Classes to Check
Structure	A named container for a hierarchical list.	`Structure`, `StructureManager`
Forest	A hierarchical list.	`Forest`, `ForestService`
Row	A row is a unique, atomic element of a forest.	`StructureRow`, `RowManager`
Item	An item is a user-level object (like Issue) that is displayed in a row.	`ItemIdentity`, `CoreIdentities`
Attribute	An attribute provides values of a certain type and meaning for forest rows.	`AttributeSpec`, `StructureAttributeService`
Column	A column loads one or more attributes and displays information about forest rows.	`ViewSpecification`
View	A view is a named collection of columns.	`StructureView`, `StructureViewManager`

Important points:

- **Structures** are the main entities provided by Structure add-on. A structure has name and other attributes, like description, and it also has content, represented by a **forest.**
- A forest represents a structure's content. But it can also represent a result of a query or a hierarchical list received or stored somewhere else.
- Forest contains **rows**. Forest content is actually a list of pairs (`row ID, depth`).
- A row has a numeric ID that uniquely identifies it in a forest. A forest may not contain the same row twice. (Although a row may be present in different forests.)
- When users look at a structure, they see a grid – each row in that grid is represented by a Structure's row.
- A row refers to an **item**. An item is an abstraction for everything that can be placed into a forest – issues, folders, projects, users are all items, from Structure's perspective.
- An item has **item identity** – something that uniquely identifies that item on a JIRA instance.
- An item also has **attributes** – some values with associated meaning, which Structure and its extensions can provide and that can be shown to the user.

10.2.2 A Note on Extensibility

Structure is built with extensibility in mind. It is possible for a separate add-on to add new item types, attributes, columns and other extensible elements to Structure, at runtime.

10.3 Accessing Structure from JIRA Plugin

Structure provides a Java API that lets other plugins interact with the Structure data. The API is accessed through a few services that you can have injected into your components.

Check out the articles below for details.

10.3.1 Setting Up the Integration

To start using Structure in your plugin:

10.3.1.1 Add dependency to your `pom.xml`

Figure out the version of the API(see page 1069) that you need – it may depend on your JIRA and Structure plugin version.

To use API classes, add the following dependency:

```
<dependency>
  <groupId>com.almworks.jira.structure</groupId>
  <artifactId>structure-api</artifactId>
  <version>17.0.0</version>
  <scope>provided</scope>
</dependency>
```

Note that there are Additional Libraries Used in Structure API(see page 1032)

10.3.1.2 Import `StructureComponents`

In your `atlassian-plugin.xml`, use `<component-import>` module to import `StructureComponents` service. This service provides access to all other Structure services.

Alternatively, you can import specific services.

```
<component-import key="structure-components" interface="com.almworks.jira.structure.api.StructureComponents"/>
```

10.3.1.3 Have Structure API service injected into your component

```
public class MyClass {
  private final StructureManager structureManager;

  public MyClass(StructureComponents structureComponents) {
    structureManager = structureComponents.getStructureManager();
  }

  ...
}
```

This is it! Continue to the list of Structure Services(see page 1035) to see which service you need to work with. Other articles in this section provide examples for specific use cases.

For a production plugin, consider Controlling Compatibility(see page 1033). For a standalone plugin, which can work without Structure, read about Making Structure Dependency Optional(see page 1034).

10.3.1.4 Additional Libraries Used in Structure API

Structure API has dependencies on a few open-source libraries that are transitively included in your project when you add a dependency on Structure API.

You don't need to explicitly add dependencies on these libraries.

Integers and HPPC

The open source library Integers[665] provides collections of primitive types with `java.util`-like interfaces. When working with `Forest`, you will typically use `LongList` and `LongArray` (an implementation of `LongList`).

It comes with another primitive type collection library, HPPC[666], which provides specific implementations of these collections.

See API Usage Samples(see page 1152) to get the idea how to work with those interfaces.

JetBrains Annotations

Annotations library from JetBrains provides `@Nullable` and `@NotNull` annotations, used throughout the API.

665 http://code.google.com/p/integers/
666 http://labs.carrotsearch.com/hppc.html

10.3.1.5 Controlling Compatibility

Why Declare Compatible Versions

Structure Java API will change with time, and it is a good practice to ensure that your plugin uses the correct version of the API.

Structure API Versions(see page 1069) page explains how version numbers change based on how compatibility is affected. Say, you develop your code using Structure API version 16.2.0 – your code will work with any version of the API starting from 16.2.0 and up to, but not including version 17.0.0.

So what happens if your code is run on JIRA with Structure that provides an incompatible API? It may break, or it may work. The exact answer depends on which parts of the API you use and what are the differences. But if the code breaks, it may not break outright – it may seem to work at first, until it tries to use a method that's not there, for example.

To make your code fail fast, you can declare dependency on a specific range of versions of the Structure API. In that case, if the version of the API is different, your plugin will fail to load and the user will immediately know that there's a problem.

Importing Specific Range of API Versions

You can declare dependency on the specific range of the API versions via OSGi bundle instructions added to your `pom.xml` or `atlassian-plugin.xml`. Figure out the compatible OSGi versions range from the API versions(see page 1069) table and modify your `pom.xml` to contain the following:

```
<plugin>
  <groupId>com.atlassian.maven.plugins</groupId>
  <artifactId>maven-jira-plugin</artifactId>
  ...
  <configuration>
    <instructions>
      <Import-Package>
        com.almworks.jira.structure.api*;version="[16,17)",
        com.almworks.integers*;version="0",
        org.jetbrains.annotations;version="0"
      </Import-Package>
    </instructions>
  </configuration>
</plugin>
```

Here we are declaring the acceptable range of versions for the Structure classes, taken from the example above. We don't much care about the versions of Integers and Annotations libraries, so `version="0"` will match any version of those packages.

You may have other `Import-Package` instructions to declare dependency rules for other packages, and you may have other instructions besides `Import-Package` as well. See the API Usage Samples(see page 1152) for a more complete example.

Next: Making Structure Dependency Optional(see page 1034)

10.3.1.6 Making Structure Dependency Optional

If you are integrating your plugin with Structure, or when you generally write code that uses Structure API but also should work when Structure Plugin is not present, you need to declare that dependencies are optional and isolate dependencies in the code.

Declare Optional Dependency

Since your plugin must first be loaded as an OSGi bundle, it should declare dependencies from the Structure API packages as optional.

Modify `<Import-Package>` declaration in your `pom.xml` or `atlassian-plugin.xml` and add `resoltion:=optional` classifier. (Add Import-Package to control API compatibility(see page 1033) if you don't have this declaration yet.)

```
<Import-Package>
  com.almworks.jira.structure*;version="[16,17)";resolution:=optional,
  com.almworks.integers*;version="0";resolution:=optional,
  org.jetbrains.annotations;version="0";resolution:=optional
</Import-Package>
```

Isolate Dependencies in the Code

So once you have declared the optional resolution of the Structure API classes, your bundle will load - but if your code tries to access a class from the Structure API, you'll get a `NoClassDefFoundError`. To avoid that, you need to isolate the dependency on Structure API classes - typically in some wrapper classes.

This is also a point to make design decisions. So your code can use Structure when it's present, and can work independently when Structure is not there. Are there any abstractions that address both of these situation? What are the concepts that are realized through Structure API and through some other means when Structure is not avialable?

Here's a sample wrapper for the Structure API that provides `ForestAccessor` wrapper (whatever it does) when Structure is available and `null` otherwise.

```
public class StructureAccessor {
  public static boolean isStructurePresent() {
    if (!
ComponentAccessor.getPluginAccessor().isPluginEnabled("com.almworks.jira.structure"))
{
      return false;
    }
    try {
      Class.forName("com.almworks.jira.structure.api.StructureComponents");
    } catch (Exception e) {
      return false;
    }
    return true;
  }

  public static ForestAccessor getForest(long structureId) {
    if (!isStructurePresent()) return null;
    StructureComponents structureComponents;
    try {
      structureComponents =
ComponentAccessor.getOSGiComponentInstanceOfType(StructureComponents.class);
    } catch (Exception e) {
      return null;
    }


    try {
      return new

ForestAccessor(structureComponents.getForestService().getForestSource(ForestSpec.stru
cture(structureId)));
    } catch (StructureException e) {
      return null;
    }
  }
}
```

10.3.2 Structure Services

This page lists public services provided by Structure API. All these services are available from StructureComponents[667] instance.

667 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/StructureComponents.html

10.3.2.1 Services to Start With

Use ...	to ...
StructureManager[668]	Create and delete structures, modify structure properties such as name or permissions. (But not to work with the structure's content.)
ForestService[669]	Access forests for reading or changing.
StructureAttributeService[670]	Retrieve attribute values for given rows in a given forest.
RowManager[671]	Extract item information for rows read from a Forest.
FolderManager[672]	Create folders or change folder properties.
GeneratorManager[673]	Create generators or change generator properties.

10.3.2.2 More Power

Use ...	to ...
StructureConfiguration[674]	Change global Structure add-on configuration.
StructureViewManager[675]	Create and manipulate views.
StructureSyncManager[676]	Manage synchronizers.
StructureBackupManager[677]	Backup complete Structure data to a file or restore it back.
StructureFavoriteManager[678]	Read or change which structures are favorite of which users.

668 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/structure/StructureManager.html
669 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestService.html
670 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/StructureAttributeService.html
671 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/row/RowManager.html
672 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/folder/FolderManager.html
673 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/generator/GeneratorManager.html
674 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/settings/StructureConfiguration.html
675 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/view/StructureViewManager.html
676 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/StructureSyncManager.html
677 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/backup/StructureBackupManager.html
678 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/structure/favorite/StructureFavoriteManager.html

Use ...	to ...
PropertyService[679]	Store arbitrary properties.
StructurePropertyService[680]	Store arbitrary per-structure properties.
AttributeSubscriptionService[681]	Create a subscription for a set of attributes and rows and load data for them asynchronously.

10.3.2.3 Extreme Power

Use ...	to ...
ItemTracker	Track recorded changes that happened to items (in JIRA Data Center – on all nodes of the cluster).
ItemResolver	Convert `ItemIdentity` into an object representing that item.
IssueEventBridge	Listen for or report issue events.
StructureQueryParser	Parse an S-JQL query.
StructureQueryBuilderFactory	Build an S-JQL query via Builder pattern.
ProcessHandleManager	Manage feedback page for asynchronous processes.
SyncAuditLog	Access or manage Synchronization Audit log.
StructureJobManager	Run a job asynchronously.
ScheduledJobManager	Schedule a periodical job to run asynchronously (only on a single node in a cluster).

10.3.3 Building Forest Specification

A forest specification, or `ForestSpec`[682], is a way for your code to identify the forest that you'd like to access. The forest may come from different sources – it could be a structure, it could be a transformed(see page 216) structure, it could be a result of query or some other types of forest source.

679 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/property/PropertyService.html
680 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/property/StructurePropertyService.html
681 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/subscription/AttributeSubscriptionService.html
682 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestSpec.html

So the first step before you read or update a forest is to create an instance of ForestSpec. Here are some examples of how you can do that.

Desired forest	ForestSpec expression
Base Content	
Structure #123	`ForestSpec.structure(123)`
Result of a JQL query	`ForestSpec.sQuery("jql", "priority = Blocker")`
Result of a text query	`ForestSpec.sQuery("text", "text to find")`
Adjusted Content	
Structure #123, sorted by Priority	`ForestSpec.structure(123).transform( CoreStructureGenerators.SORTER_ATTRIBUTE, ImmutableMap.of( "attribute", (Object) ImmutableMap.of("id", IssueFieldConstants.PRIORITY, "format", "order") "desc", true ) );`
Structure #123, skeleton only (without dynamic content)	`ForestSpec.skeleton(123)`
Structure #123, with title row	`ForestSpec.skeleton(123).withTitle()`

More details are available in Javadocs for ForestSpec[683].

683 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestSpec.html

10.3.4 Reading Structure Content

Let's say you need to access a structure's content and export the hierarchy into your custom format or use for displaying the hierarchy in your way. This scenario walks you through from having just a structure name to iterating through the forest and learning which items are there.

We assume that your code has `StructureComponents` instance injected into `myStructureComponents` field.

10.3.4.1 Figure out Structure ID

To address a structure, you need to know its ID. If you just have a name you can do the following:

```
List<Structure> structures =
myStructureComponents.getStructureManager().getStructuresByName("My Structure",
PermissionLevel.VIEW);
long structureId;
if (structures.size() == 1) {
  structureId = structures.get(0).getId();
} else {
  // no structures or too many structures -- error?
}
```

Now you have `structureId` or an error situation where the name does not uniquely identify your structure.

10.3.4.2 Create a ForestSpec

You need a forest specification to get a ForestSource. You can read more about this in the section about Building Forest Specification(see page 1037).

```
ForestSpec forestSpec = ForestSpec.structure(structureId);
```

Note that this forest spec is going to be "secured" for the current user, which means that the resulting forest will exclude the sub-trees that only contain items not visible to the user.

10.3.4.3 Retrieve ForestSource

A `ForestSource` is an interface that produces some specific forest and that provides versioning for it.

```
ForestSource forestSource =
myStructureComponents.getForestService().getForestSource(forestSpec);
```

Note that this call may produce `StructureException` in case a structure cannot be found and in some other cases. A robust code would have some exception handling.

⚠ Do not store a `ForestSource` in memory for a long time, longer than a single user request. Structure has internal caching engine that efficiently manages forest sources and their dependencies. Request forest source from `ForestService` in every new request.

10.3.4.4 Retrieve Forest and its version

Forest source can provide you with the latest version of the forest, or with an incremental update, based on the version you already have.

To get the latest forest:

```
VersionedForest versionedForest = forestSource.getLatest();
DataVersion latestVersion = versionedForest.getVersion();
Forest forest = versionedForest.getForest();
```

Note that `latestVersion` variable contains the version of the forest that you got. You can later use it to call `forestSource.getUpdate(latestVersion)` and receive only information about how did the forest change since the last time you've seen it.

⚠ You cannot really use `latestVersion` for anything else besides getting updates later. The numbers in that version bear no meaning regarding structure's history. For history queries, you'll need to use `HistoryService`.

10.3.4.5 Iterate through Forest and get StructureRow instances

A `Forest` is just two parallel arrays, one containing row IDs, the other containing depths. (Or, one can say that it is a list of pairs `(rowId, depth)`.) You can iterate through it via simple cycle.

For each row, you'll need more information than just row ID. We use `RowManager` to retrieve other properties of a row.

```
RowManager rowManager = myStructureComponents.getRowManager();
for (int i = 0; i < forest.size(); i++) {
  long rowId = forest.getRow(i);
  int depth = forest.getDepth(i);
  StructureRow row = rowManager.getRow(rowId);
  ...
}
```

Note that `row` is never `null`, because Row Manager would through an unchecked exception if a row is not found – this situation is considered a developer's error.

10.3.4.6 Analyze the row and process data

Finally, you get `ItemIdentity` from the row to understand which item does the row show. The items could be anything – issues, folders, users. So even if your structure only contains issues, it is advised to do an extra check.

```
ItemIdentity itemId = row.getItemId();
if (CoreIdentities.isIssue(itemId)) {
  long issueId = itemId.getLongId();
  // process the row!
  ...
}
```

A structure with dynamic content will also contain generators. If you take all the rows, regardless of the item type and use them somewhere, you might stumble upon a generator. To eliminate them from the analyzed forest, add a condition. The same is usually done for "loop markers", which are special items added by extenders to indicate that there's a loop (like cyclic issue links).

```
  ItemIdentity itemId = row.getItemId();
  if (!CoreIdentities.isGenerator(itemId) && !
CoreIdentities.isLoopMarker(itemId)) {
    ...
  }
```

Congratulations! You've successfully implemented forest read-out.

You can adjust this walkthrough for your needs – for example, read a query result, or read only a portion of a forest.

10.3.5 Changing Structure Content

Updating a structure can be done through the same `ForestSource` interface that was used for Reading Structure Content(see page 1039). In this article, we're assuming that you've got `forestSource` local variable that you've created according to instructions in the previous article.

10.3.5.1 Forest Coordinates

To make a change to a forest, you need to be able to point to a specific part of a forest. This is done by using row IDs, which uniquely identify forest rows.

- To point to a specific row in the forest, which you'd like to move or delete, you just use this row's ID.
- To point to a specific position in the forest, where you'd like to insert or move rows to, you need to use row IDs of its neighbors, or *coordinates*:
  - "Under" coordinate is the row ID of the future parent of the inserted row, or zero if the row is placed at the top level.
  - "After" coordinate is the row ID of the future preceding sibling of the inserted row under the same parent, or zero if the row is placed as the first child.
  - "Before" coordinate is the row ID of the future succeeding sibling of the inserted row under the same parent, or zero if the row is placed as the last child.

10.3.5.2 Applying Forest Action

To make a change, you need to call `forestSource.apply()` method, passing a specific `ForestAction` that you want to apply.

Adding a single row

To add a single row to the forest, use `ForestAction.Add` constructed with the `ItemIdentity` of the item associated with that row.

```
forestSource.apply(new ForestAction.Add(CoreIdentities.issue(10000), under, after,
before))
```

Adding a sub-forest

To add multiple rows in one action, use `ForestAction.Add` that receives an `ItemForest`.

`ItemForest` is a special container that is used to build a temporary forest with temporary rows, having negative row IDs. The class provides information both about the hierarchy of inserted temporary rows (via `Forest`) and a mapping from the temporary row ID to the inserted `ItemIdentity`.

To create an `ItemForest`, you need to use either `ImmutableItemForest` or `ItemForestBuilderImpl`.

```
ItemForest itemForest = new ItemForestBuilderImpl()
  .nextRow(CoreIdentities.textFolder("My Issues"))
  .nextLevel()
    .nextRow(CoreIdentities.issue(10000))
    .nextRow(CoreIdentities.issue(10001))
    .build();
forestSource.apply(new ForestAction.Add(itemForest, under, after, before));
```

Removing a sub-tree

To remove a row, use `ForestAction.Remove` and pass the row ID being removed.

```
forestSource.apply(new ForestAction.Remove(LongArray.create(100, 101, 102)));
```

⚠ All sub-rows of the removed rows will be removed as well. If you need to keep them, apply `ForestMove` on them first.

Moving a sub-tree

To move a row with its sub-rows, use `ForestAction.Move`.

You can specify one or more row IDs, which can be from the different parts of the forest. Those rows will be placed one after another at the specified position.

```
forestSource.apply(new ForestAction.Move(LongArray.create(100, 101, 102), under,
after, before));
```

10.3.5.3 Inspecting the Results

A call to `ForestSource.apply()` will finish successfully if the operation has been completed and throw a `StructureException` otherwise.

You can inspect the returned `ActionResult` to get information about the *effects* of the action (more on effects below).

You can also use `ActionResult.getRowIdReplacements()` – it is a mapping from the temporary row IDs, used when adding rows, to the newly assigned real row IDs, which are now part of the structure.

10.3.5.4 Effects and Changing Dynamic Structures

You may have noticed that you can apply actions to any forest source, not necessarily a simple structure. It can be a transformed structure, or even a transformed query. A structure can also contain dynamic parts, created or adjusted by generators, and you can try to apply the actions that would affect these parts.

A successful action would produce one or more *Effects* (represented in the `ActionResult` as `AppliedEffect`). In simple case of changing a non-dynamic structure, it would be, unsurprisingly, a structure change. In case the action involves dynamic content, the effects may differ – but the general concept is that, after the effect takes place, the updated (re-generated) structure will reflect the desired action's result.

Here are some examples of the possible effects.

Action	Effect
Adding rows to a static structure	✅ Structure is modified
Moving item X from group A to group B, where groups are provided by a grouper by field F	✅ The value of F for X is changed from A to B
Removing issue X from under issue Y, when previously X was added automatically by a Links Extender using link type L	✅ Link L: Y → X is deleted
Moving issue upwards when structure is sorted by Agile Rank	✅ Issue's Rank is changed
Adding an issue to an arbitrary JQL query result	⛔ StructureException is thrown – no way to force an issue to be part of a JQL result
Adding issue X under issue Y within the scope of a Links Extender and when issue Y is "static" (not added by the extender)	⛔ StructureInteractionException is thrown – there are two ways to interpret this action

As generators are extensible and can be added by other plugins, the range of possible effects is not limited.

Note that in the last two examples the action is not successful. In the last example, you need to use `ForestSource.apply()` with parameters, which would define whether a generator should process the action or if the issue should be inserted into the static structure.

10.3.5.5 Concurrency and Atomicity

Each `ForestAction` can be viewed as a separate transaction. It is atomic, meaning that it is either fully successful or fully failed.

There's no way to make a transaction larger. In other words, if you apply two actions to a forest source, it is possible that a concurrent action, done from another thread, is executed in between your two actions.

10.3.5.6 Permissions

All actions are executed under the "current" user and with all necessary permission checks. Updating a structure requires `EDIT` permission on the structure. Other effects, like changing issue fields, would require `EDIT_ISSUE` permission on the subject issues.

When permissions are insufficient, the action will not succeed and a `StructureException` will be thrown.

> When it comes to effects applied by generators, it is a generator's responsibility to check permissions before applying an action. All generators bundled with Structure have strict permission checks.

The current user is generally managed by JIRA and is the same as the user who makes the request. However, you can use `StructureAuth` class to "sudo" to another user or to bypass permission checks altogether.

10.3.6 Loading Attribute Values

You may need to load the same values that Structure shows on the Structure Board, especially if it's a total value, progress value or other Structure-specific value. This is done via `StructureAttributeService`.

10.3.6.1 About Attributes

One of the core concepts in Structure is the Attribute abstraction. An attribute is something that can provide a value of specific type and meaning for any row in a forest.

For example, a "Summary" attribute would produce the value of Summary field for issues, the name of a folder for folders and a person's full name for users. Some attributes may be applicable only to certain item types and would provide empty value for all other items.

Besides item-based attributes, which provide values that depend only on the item in the forest, there are forest-based attributes, which are calculated based on the whole forest and items in it.

> Forests and Attributes are two main concepts that make up the Structure grid. Looking at the Structure Board, you see Forest in the vertical direction – rows and hierarchy are taken from Forest, and you see Attributes in the horizontal direction – all columns load Attributes from the server and display those values.

10.3.6.2 General Approach to Loading Values

Let's assume that, after Reading Structure Content(see page 1039), you have `StructureComponents` instance and an instance of `ForestSpec` for a forest. We can read a number of attributes for a number of rows by going to `StructureAttributeService`.

Figure out which Attributes do you need

The service accepts multiple attribute specs in one request. If you need several attributes calculated – it's better to do that in one request.

```
List<AttributeSpec<?>> attributeSet = new ArrayList<>();
attributeSet.add(CoreAttributeSpecs.KEY);
attributeSet.add(CoreAttributeSpecs.SUMMARY);
attributeSet.add(CoreAttributeSpecs.TOTAL_REMAINING_ESTIMATE);
```

`CoreAttributeSpecs` class and its parent class, `SharedAttributeSpecs`, contain some of the most popular attributes.

It's likely that you'll need to build you own attribute specification. For example, to address a numeric JIRA custom field and calculate total of that field based on sub-issues, you'll need the following.

```
AttributeSpec<Number> customField =
  AttributeSpecBuilder.create("customfield",
ValueFormat.NUMBER).params().set("fieldId", 10000).build();

AttributeSpec<Number> customFieldTotal =
  AttributeSpecBuilder.create(CoreAttributeSpecs.Id.SUM,
ValueFormat.NUMBER).params().setAttribute(customField).build();

attributeSet.add(customFieldTotal);
```

Figure out which Rows do you need to calculate the Attributes for

For example, this could be all rows in that structure.

```
LongList rows =
myStructureComponents.getForestService().getForestSource(forestSpec).getLatest().getF
orest().getRows();
```

If you need to create a `LongList` manually, use `LongArray` implementation.

Call `StructureAttributeService`

This service calculates a matrix of values for each row and attribute you specify.

```
RowValues values =
myStructureComponents.getAttributeService().getAttributeValues(forestSpec, rows,
attributeSet);
```

There is a variation of `getAttributeValues()` method that accepts a `Forest`, rather than `ForestSpec`. It is recommended to use the variant that accepts `ForestSpec` whenever possible, because that variant uses caching.

Read out the result

The returned object contains values for all pairs of requested row and requested attribute.

```
for (LongIterator ii : rows) {
  String key = values.get(ii.value(), CoreAttributeSpecs.KEY);
  Number total = values.get(ii.value(), customFieldTotal);
  ...
}
```

10.3.7 Creating and Adding Folders

You may need to create a new folder and add it to a structure.

Folders and generators are items that are managed entirely by Structure add-on, so you'll need to use Structure's services to create the item first, giving you the item identify, and then insert a row into a forest.

Read more about Changing Structure Content(see page 1041) for general ideas about updating a structure.

10.3.7.1 Create the Folder entity

```
long folderId = myStructureComponents.getFolderManager().createFolder(Folder.named("M
y Stuff").build());
```

The folder is now stored in the database.

10.3.7.2 Define folder's identity

```
ItemIdentity itemId = CoreIdentities.folder(folderId);
```

10.3.7.3 Add folder to structure

```
forestSource.apply(new ForestAction.Add(itemId, 0, 0, 0));
```

10.3.8 Creating Dynamic Structures

Structures may have dynamic content, produced by generators.

Generators can be added to structure and moved around in the same way other items are added, as described in Changing Structure Content(see page 1041). A generator will have effect on the whole sub-tree under its parent.

Generators are a separate entities, managed by Structure add-on. So to create a dynamic structure, we need to create a generator first and then insert it into the structure.

10.3.8.1 Create generator instance

You create a generator instance by calling `GeneratorManager`.

```
long generatorId = myStructureComponents.getGeneratorManager().createGenerator(
  CoreStructureGenerators.SORTER_AGILE_RANK,
  ImmutableMap.of(CoreGeneratorParameters.SORT_DESCENDING, false),
  structureId);
```

Note the third parameter – the generator is "owned" by a structure, so we should pass the ID of the owning structure.

10.3.8.2 Insert generator into the forest

Find parent row under which you'd like the forest to be automated. To apply generator to the whole forest insert generator at the top level by making "under" coordinate zero.

Do not use "after" and "before" coordinates unless you are adding an Inserter.

```
forestSource.apply(new ForestAction.Add(CoreIdentities.generator(generatorId), under,
0, 0));
```

This is it! Next time you read the contents of this forest source, it will have the results of this generator applied.

10.4 Extending Structure Functionality

You can extend Structure add-on's functionality with your own add-on by using one of the available extension points.

- Creating a New Column Type(see page 1048)
- Loading Additional Web Resources For Structure Widget(see page 1065) — To include a web resource (such as custom CSS or JavaScript file) on the page every time Structure Widget is displayed, use structure.widget web resource context and possibly a few others.
- Declaring a New Generic Item Type(see page 1066)

Structure plugin has a lot of extension points. More extensive documentation is coming with the future versions. It will cover the following topics:

- Adding new item types, which can be used in a structure
- Adding new generators, which can build dynamic structures (Inserters, Extenders, Filters, Groupers and Sorters)
- Adding new attributes, displaying them in the Structure grid or using for sorting or grouping
- Adding new structure templates
- Adding new constraint function to S-JQL
- Adding actions to Manage Structure page
- Adding toolbar elements to the Structure Board

If you're interested in these topics but cannot find documentation or need help, please contact Tempo Support[684] and we'll provide advice.

10.4.1 Creating a New Column Type

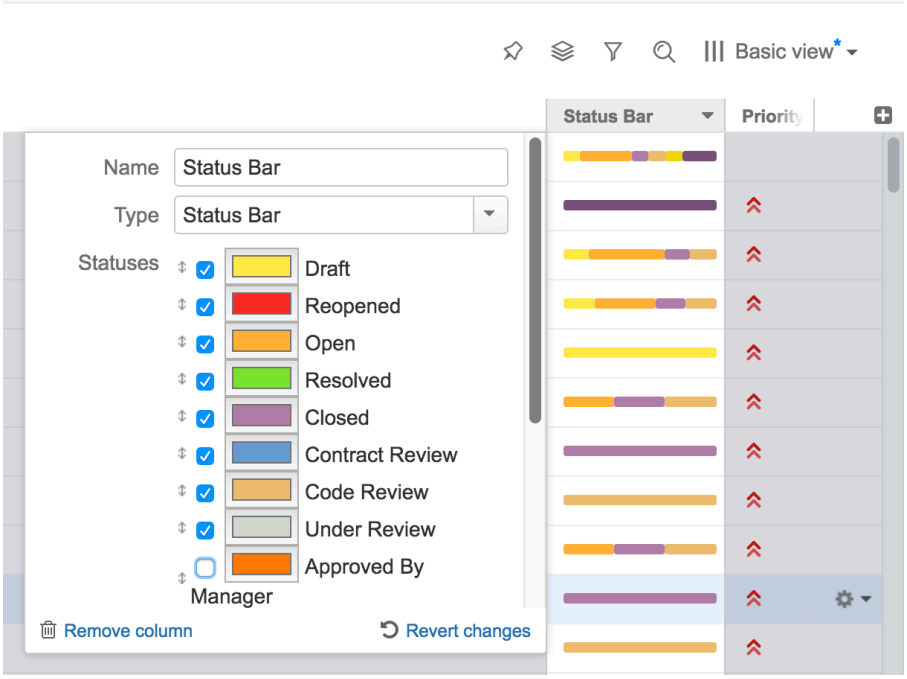
In this tutorial we will develop the Status Bar column type, which shows a progress-like bar filled with color stripes, each stripe's color representing a particular issue status, and each stripe's width being proportional to the number of issues having that status in the current issue's subtree.

You can download both the compiled plugin and its source code from API Usage Samples(see page 1152).

- The Plan(see page 1049)
- The Attributes(see page 1049)
- AttributeSpec for Status Bar(see page 1050)
- Status Bar Attribute(see page 1050)
- Attribute Provider(see page 1052)
- Client-Side Column(see page 1053)
  - API Overview(see page 1053)
  - Column Specifications(see page 1054)
  - The Column Context(see page 1055)
  - Requesting and Using Metadata(see page 1057)
  - Column(see page 1058)
  - ColumnConfigurator(see page 1059)
  - ColumnOption(see page 1059)
  - ColumnType(see page 1061)
  - Column Groups(see page 1062)
  - Web Resources and Contexts(see page 1062)
- Export Renderers(see page 1062)

684 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

- Export Strategies(see page 1063)
- Generic Renderer Provider(see page 1064)
- Advanced Excel Renderer Provider(see page 1064)



10.4.1.1 The Plan

A column type consists of several components. The client-side components are written in JavaScript and have two responsibilities:

- Rendering the cells in the Structure widget.
- Providing the column configuration UI.

The server-side components are written in Java and responsible for:

- Providing the attributes needed by the client-side part to render the cells.
- Exporting the column into printable HTML and Microsoft Excel formats.

For the Status Bar column we'll need to write code to cover all of the above responsibilities.

In general, however, only the client-side part is strictly necessary. If the attributes provided by Structure(see page 0) are enough for your column, you can skip the server-side attribute provider. You can also skip the components related to export, if this functionality is not critical. In that case, you can jump straight to the client-side part(see page 1053), consulting the other chapters as necessary. For the complete treatment, please continue reading from top to bottom.

10.4.1.2 The Attributes

Before we begin, let's decide which attributes we need to pass from the server side to render a status bar. Obviously, the status bar depends on the statuses of all the issues in the given issue's subtree. This suggests that we need to use an "aggregate" attribute, and because Structure does not provide such an aggregate out of the box, we'll need to write our own.

Secondly, the colors and the order of statuses in the status bar are only a presentational matter. If we had a map from status IDs to sub-issue counts in the given issue's subtree, we could count the total number of sub-issues, scale the colored stripes so that they'd fill the whole status bar, and render them in any given order.

Thirdly, the "Include itself" option is somewhat trickier. When it's on, the current issue's status is shown in its status bar, as if there is one more sub-issue. When it's off, the current issue is excluded, and the status bar shows only its sub-issues (on all levels). We could try to implement this on the server side as a separate aggregate, however, this approach has a couple of drawbacks:

- When the user toggles the checkbox, Structure will have to calculate a new aggregate and transfer the results. Because the aggregate values are cached on the server side, and issue data values are cached on the client side, on both sides we'll have increased memory consumption.
- Because of the way the aggregates are calculated and cached on the server side, the aggregate for the option turned off will be somewhat more difficult to write, and use a more complex data structure.

So, we'll do things differently, and use a single, simpler, aggregate, calculating the data with the "Include itself" option turned on. If it's off, we'll adjust the data on the client side. To do that, we'll need another piece of data – the status ID for the current issue, but that can be provided by Structure itself, and the overhead of requiring it is less than that of a separate aggregate.

10.4.1.3 AttributeSpec for Status Bar

Once we understood which attributes will our JavaScript code need, we have to define or find the appropriate attribute specifications for it.

Our status bar is going to be a new attribute, so we need to create an AttributeSpec[685]. The ID for this spec should be something unique to our add-on. And the format should be a generic `JSON_OBJECT`, because we're going to transfer a bunch of data back to the client rather than just a single value.

```
public static final AttributeSpec<Map<String, Integer>> STATUS_BAR
  = new AttributeSpec("com.almworks.statusbar", ValueFormat.JSON_OBJECT);
```

We don't need any parameters for this attribute specification – regardless of column configuration, we'll always load the same attribute.

The value will be the map from the Status ID to the number of cases that status is encountered in the sub-tree, including the parent issue.

As for the status ID of the current row, we'll use `CoreAttributeSpecs.STATUS_ID`.

10.4.1.4 Status Bar Attribute

Now that we know which attribute we need to implement, let's write a loader of that attribute. A loader is an instance of AttributeLoader[686] that loads specific attributes for a specific request.

We need to start by looking for the most convenient base class for our loader. It seems that `AbstractNaiveDistinctAggregateLoader` is the best, because:

- It is already a loader for an aggregate,
- It addresses the problem of having multiple issues in the same sub-tree more than once – obviously, we don't want to count such issue's Status twice.

685 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/AttributeSpec.html
686 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/loader/AttributeLoader.html

This basic class is "naive" because the way it implements the distinct counting is very basic – and potentially consuming a lot of memory. Structure has more advanced methods for calculating the distinct totals, but it's not a part of the API as of version 17.

As the loader does not have any other parameters, we'll only need a single instance, which we'll keep in a `static final` field.

```
private static final AttributeLoader<Map<String, Integer>> LOADER = new
 StatusBarLoader();
```

Our loader will have a dependency on the `CoreAttributeSpecs.STATUS` attribute. Structure will guarantee that the dependency attributes are loaded before our loader is asked to do its calculation.

All multi-row attribute loaders (aggregates, propagates and scanning loaders) must not access items directly, but rather declare dependency on item-based or single row-based attributes. This is needed by the attributes subsystem to function efficiently, but also, it lets other developers extend the applicability of those dependency attributes to a new type of items. In that way your aggregate attribute will immediately work with the new item types, even though you didn't know about them at development time.

The calculation of the result is pretty straightforward. The base class, `AbstractDistinctAggregateLoader`, defines two methods for building recursive value: `getRowValue()` provides a single value for a single row and `combine()` accumulates the provided values.

- As a result for a single row, we create a map with just one record: the issue's status is mapped to 1. If status is missing (as would be the case for non-issues), we just return null.
- As a combination function we will implement map merge that combines counters.
- Finally, we return an immutable copy of the `result` map.

StatusBarAggregate.java

```
  private static class StatusBarLoader extends
 AbstractNaiveDistinctAggregateLoader<Map<String, Integer>> {
    public StatusBarLoader() {
      super(STATUS_BAR);
    }

    public Set<AttributeSpec<?>> getAttributeDependencies() {
      return Collections.singleton(STATUS);
    }

    protected Map<String, Integer> getRowValue(RowAttributeContext context) {
      Status value = context.getDependencyValue(STATUS);
      return value == null ? null : Collections.singletonMap(value.getId(), 1);
    }

    protected Map<String, Integer> combine(Collection<Map<String, Integer>> values,
AggregateAttributeContext context) {
      HashMap<String, Integer> r = new HashMap<>();
      for (Map<String, Integer> map : values) {
        if (map != null) {
          for (Map.Entry<String, Integer> e : map.entrySet()) {
            Integer count = r.get(e.getKey());
            if (count == null) {
              count = 0;
            }
            r.put(e.getKey(), count + e.getValue());
          }
        }
      }
      return r;
    }

  }
```

10.4.1.5 Attribute Provider

Attribute providers are registered as modules in the plugin descriptor, and their instances are created by the JIRA module system. If the attribute provider "recognizes" the attribute specification and can serve it, it must return a non-null `AttributeLoader` instance. Because our `StatusBarLoader` implementation is stateless and has no parameters, we can reuse the single `static final` instance, but a configurable data provider could create and return new loaders for each call. The returned loader will then be called once for each item needed to display the Structure grid (or its visible part).

StatusBarDataProvider.java

```
public class StatusBarAttributeProvider implements AttributeLoaderProvider {
  private static final AttributeSpec<Map<String, Integer>> STATUS_BAR = new
 AttributeSpec("com.almworks.statusbar", ValueFormat.JSON_OBJECT);
  private static final AttributeLoader<Map<String, Integer>> LOADER = new
 StatusBarLoader();

  public AttributeLoader<?> createAttributeLoader(AttributeSpec<?> attributeSpec,
@NotNull AttributeContext context)
    throws StructureProviderException
  {
    if (STATUS_BAR.getId().equals(attributeSpec.getId())) {
      return LOADER;
    }
    return null;
  }
}
```

When the data provider is ready, we register it in the plugin descriptor.

atlassian-plugin.xml

```
<structure-attribute-loader-provider key="alp-sbcolumn" name="attribute-loader:Status
Bar Column"
                                     class="com.almworks.jira.structure.sbcolumn.Stat
usBarAttributeProvider"/>
```

10.4.1.6 Client-Side Column

We now come to the most visible part of the column – the client-side JavaScript code, responsible for rendering the cells of the Structure grid and showing the column configuration UI. Having almost 400 lines of JavaScript, the code is too long to be reproduced in its entirety. We advise you to download the API examples source code from the API Usage Samples(see page 1152) page and open `sbcolumn.js` from the `status-bar-column` sample plugin in your favorite editor.

First, we'll take a high-level overview of the API and look at a few common concepts – column specifications, column context, and the metadata. After that we'll discuss each of the API classes and their implementations.

API Overview

The whole API is accessible through the `window.almworks.structure.api`(see page 1132) global object. There are a few utility functions and four main classes that the developer needs to extend (by using the `api.subClass()` function) in order to create a fully-functional column. These classes are linked together by the **column specification**, which is a JSON object representing all of the column's parameters. Column specifications are discussed in detail in the following section. Now let's overview the classes and functions.

Class or Function	Description
`api.ColumnType`(see page 1144)	The **column type** is the gateway between Structure and your code. The column type is registered with Structure and has the following responsibilities: • creating column presets for the "Add Column" menu; • creating the column preset used when switching to your column type from a different type; • creating `Column` and `ColumnConfigurator` instances for given column specifications.
`api.Column`(see page 1135)	The **column** is responsible for value rendering. It creates the HTML for the widget cells and controls the column's name and width. It can require one or more attributes to be downloaded from the server for the rendered rows.
`api.ColumnConfigurator`(see page 1139)	The **configurator** is responsible for the column configuration panel as a whole. Its most important task is to create `ColumnOption` instances.
`api.ColumnOption`(see page 1141)	The **option** is the workhorse of the configuration UI, corresponding to a single "row" of the configuration panel. It is responsible for creating the input elements and routing changes between them and the specification.
`api.registerColumnType(columnType, columnKey)`	Registers a column type with the Structure, making it responsible for handling the given column key (see below).
`api.registerColumnGroup(parameters)`	Registers a new group in the Add Column menu.

Column Specifications

A **column specification** is a JSON object representing the complete configuration of a Structure widget column. Column specifications are stored as parts of view specifications. Each `Column`, `ColumnConfigurator` and `ColumnOption` instance has its own current specification, accessed via `this.spec`. A `ColumnType` is given a column specification when Structure wants it to create a `Column` or a `ColumnConfigurator`. `ColumnType` also creates column specifications for column presets. Finally, column specifications are passed to the export renderer providers on the server side (see below).

Do not confuse column specifications with attribute specifications. A column is a higher-level concept and may require multiple attributes (as is the case with our Status Bar column).

Here is an example of a Status Bar column specification.

```
{ "csid":   "7",
  "key":    "com.almworks.jira.structure.sbcolumn",
  "name":   "Status Bar",
  "params": {
    "statuses":      ["1", "3", "4", "5", "6", "10000"],
    "colors":        ["#fcaf3e", "#fce94f", "#ef2929", "#8ae234", "#ad7fa8",
"#729fcf"],
    "includeItself": true }}
```

Key	Description
`csid`	The **CSID** ("column sequential ID") is a string that uniquely identifies a column within a view. CSIDs are assigned and managed by Structure, and should not bother you as a column developer. **Do not change a column's CSID!**
`key`	The **key** is a string identifying the column type. Structure uses the key to decide which `ColumnType` or `ExportRendererProvider` to use for a particular column. The key is required.
`name`	The column **name** is shown in the column header. The name is often omitted from the specification, in which case a default name is generated for the column.
`params`	This is a JSON object containing the column's parameters. The layout of this object is up to the column developer. In the example we see two parallel arrays for the selected status IDs and their colors, and a `boolean` for the "Include itself" option.

The Column Context

A **column context** is a JavaScript object providing various kinds of information about the environment, in which columns and their configurators operate. It is not to be confused with the somewhat similar in purpose, but unrelated `AttributeContext` on the server side. When Structure makes requests to the `ColumnType`, it passes the context as a parameter. Each `Column`, `ColumnConfigurator` or `ColumnOption` instance has its own current context, accessed via `this.context`. The table below describes the methods of the column context.

Method	Description
`structure.isPrimaryPanel()`	Returns `true` if the column belongs (or will belong, for presets) to the primary panel of the Structure widget.

Method	Description
`structure.isSecondaryPanel()`	Returns `true` if the column belongs (or will belong, for presets) to a secondary panel(see page 518) of the Structure widget.
`structure.isStructureBoard()`	Returns `true` if the current widget is on the Structure Board page.
`structure.isIssuePage()`	Returns `true` if the current widget is in the Structure section of an issue page.
`structure.isGadget()`	Returns `true` if the current widget is embedded in a Structure gadget.
`structure.isLocalGadget()`	Returns `true` if the current widget is embedded in a **local** Structure gadget (i.e. a gadget provided and rendered by the same server).
`structure.isRemoteGadget()`	Returns `true` if the current widget is embedded in a **remote** Structure gadget (i.e. a gadget provided and rendered by different servers).
`structure.isGreenHopperTab()`	Returns `true` if the current gadget is in the Structure section of an Agile (GreenHopper) board.
`structure.isProjectPage()`	Returns `true` if the current gadget is in the Structure tab of a project page.
`jira.getAllIssueFields()`	Returns an array of JSON objects representing available JIRA issue fields.
`jira.getIssueFieldById(fieldId)`	Returns a JSON object representing the JIRA issue field with the given ID, or `undefined` if there is no such field.
`getMetadata(key)`	Returns the metadata object associated with the given `key`. See the section below for the description of metadata.

In our column we'll use `context.getMetadata()`.

Requesting and Using Metadata

Metadata, in the context of the column API, is any data needed by column types, columns, and configurators to to do their duties, except for attributes. For example, the Status Bar column needs to know the IDs and names of all the issue statuses in order to render tooltips and create presets – this is metadata. Structure provides some metadata by default – the `getAllIssueFields()` and `getIssueFieldById()` methods of the column context are examples, but you can load more via AJAX by issuing **metadata requests**.

Metadata is requested by overriding one or more of the methods in `ColumnType`, `Column`, and `ColumnConfigurator` classes. Let's look at an example from the Status Bar column type:

sbcolumn.js

```
getMetadataRequests: function() {
  return {
    status: {
      url: baseUrl + '/rest/api/2/status',
      cacheable: true,
      extract: function(response) {
        var result = { order: [], names: {} };
        if ($.isArray(response)) {
          response.forEach(function(status) {
            result.order.push(status.id);
            result.names[status.id] = status.name;
          });
        }
        return result;
      }
    }
  };
}
```

The method is supposed to return a JavaScript object. Each key in that object will become a metadata key for obtaining the corresponding result from the column context. In this example, the status-related metadata object will be obtained by calling `context.getMetadata('status')`.

The values in the returned object are request specifications. Let's look at the request properties:

- The `url` property is the URL to be requested. Here we call a JIRA REST API method that returns all available issue statuses. **Don't forget the JIRA base URL!**
- The `cacheable` property is an opt-in mechanism for response caching. If a metadata request is cacheable, and this URL has already been requested (e.g. by a different column type), the previous response will be used instead of making a new AJAX request. You should **declare your requests cacheable whenever possible** to conserve traffic and improve responsiveness.
- The `extract` property is the function that receives the response and produces the value stored in the metadata map. If omitted, the response is stored unchanged. In the example, we convert the resulting array of JSON objects into an array of status IDs and a map from status IDs to status names.
- You can add any other properties supported by `jQuery.ajax()` to the request specification. Remember, though, that the jQuery success and error handlers will not be called for cacheable requests if a cached response is used.

Different metadata may be required for different operations. Therefore, there are several methods in the API that you can override to request metadata:

- A column type may request metadata to be able to:
  - create column presets – `ColumnType.getPresetMetadataRequests()`;
  - create columns from specifications – `ColumnType.getColumnMetadataRequests()`;
  - create configurators from specifications – `ColumnType.getConfigMetadataRequests()`;
  - do all of the above – `ColumnType.getMetadataRequests()`, the "catch-all" method.
- A column may need metadata to render its values – `Column.getMetadataRequests()`.
- A configurator may need metadata to set up the UI – `ColumnConfigurator.getMetadataRequests()`.

Please note that the corresponding type-level metadata is also available to the columns and configurators created by the type. So, for example, there is no need to issue *the same* requests in both `ColumnType.getColumnMetadataRequests()` and `Column.getMetadataRequests()`, the former alone will suffice.

Structure will delay loading the metadata for as long as possible. For example:

- the metadata for a column will not be loaded unless there is a column in the widget that needs it;
- the metadata for creating column presets will not be loaded until the user clicks "Add Column" or "Edit Column" icons;
- and so on.

Structure guarantees that the metadata request will be completed by the time it calls your type, column, and configurator methods (obviously, except for the `getMetadataRequests()` methods themselves). If the requests succeed, the metadata will be available in the column context. If they fail, the corresponding metadata will be `undefined`, but the methods will still be called, and they should not fail in that case.

`Column`

The `api.Column` class is responsible for rendering the cells of the Structure grid. Please refer to the Column class reference(see page 1135) for the list of methods that you can override. The `StatusBarColumn` class in `sbcolumn.js` overrides four methods.

`getDefaultName()` simply returns a localized string as the column name when the name is not present in the column specification. A more involved column could use its specification, context, or metadata to determine the default column name.

`canShrinkWhenNoSpace()` allows Structure to make the column narrower than its minimum width when the widget is very low on horizontal space. Because we do not override any other sizing-related methods, the column will be resizable, with the default and minimum width of 120 and 27 pixels, respectively. Autosizing will not be applied to it, because there is no variable-size content, so autosizing makes no sense.

`collectRequiredAttributes()` always requests the status bar aggregate data from `StatusBarAttributeProvider`. If the "Include itself" option is off, it additionally requests the status ID of the current issue, which is provided by Structure as `{id:'status',format:'id'}`. The main attributes are also available from `require('structure/widget/attributes/Attributes')` object.

`getSortAttribute()` is used to specify the attribute for sorting when the user clicks on the column header.

`getCellViewHtml()` returns the actual HTML for the cells. It obtains the serialized status bar map from the `renderParameters`, deserializes it, adjusts for the "Include itself" option, if necessary, distributes the full status bar width of 100% among the selected statuses according to their issue counts, and finally generates and returns the status bar HTML code as a string. Please refer to the source code for the implementation details.

Please note, that for simple columns, displaying textual information, we advise you to override `getCellValueHtml()` instead, and let Structure take care of the boilerplate HTML surrounding your value.

However, since we want the Status Bar to look similar to Structure's Progress Bar, we need to override a higher-level method and mimic the Progress Bar HTML layout.

ColumnConfigurator

The api.ColumnConfigurator(see page 1139) class is responsible for the column configuration UI. Because most of the work is delegated to ColumnOption instances (see below), the configurators themselves are usually quite simple. Let's look at StatusBarConfigurator in its entirety.

sbcolumn.js

```
var StatusBarConfigurator = api.subClass('StatusBarConfigurator',
api.ColumnConfigurator, {
  init: function() {
    this.spec.key = COLUMN_KEY;
    this.spec.params || (this.spec.params = {});
  },
  getColumnTypeName: function() {
    return AJS.I18n.getText("sbcolumn.name");
  },
  getGroupKey: function() {
    return GROUP_KEY;
  },
  getOptions: function() {
    return [new StatusesOption({ configurator: this }), new
 IncludeItselfOption({ configurator: this })];
  }
});
```

The constructor, init() simply sanitizes the current column specification.

getColumnTypeName() returns the human-readable name for the column type. This name is used in the "Type" drop-down of the column configuration panel. You can also override getDefaultColumnName() to generate column names if the type name cannot always be used as the default column name.

getGroupKey() returns the key of the group in the "Add Column" menu that will contain this preset. See the sections on ColumnType(see page 1061) and column groups(see page 1062) below.

getOptions() creates and returns an array of ColumnOption instances that create input controls for the column configuration panel and route events. Please note how the configurator instance is passed to each option's constructor – this is crucial. The order of the options in the resulting array is also important – the rows of the configuration panel will be created in that order.

Although the methods of StatusBarConfigurator always return the same values, this is not a requirement. The result of any of the methods can depend on the current column specification (this.spec) and metadata.

ColumnOption

Each api.ColumnOption(see page 1141) instance is responsible for editing a single logical "part" of the column specification, and corresponds to a single "row" of the column configuration panel. The option creates the actual input elements and sets up event handlers to transfer the values between the inputs and its column specification. An option can hide itself if it's not applicable to the current specification. Also, each option can prohibit saving the

column configuration if it considers the current specification invalid – see `isInputValid()` method in the class reference.

Status Bar column has two options:

- `StatusesOption` is responsible for status selection, colors, and ordering. It "owns" the `statuses` and `colors` arrays of a Status Bar column specification. This option is somewhat more involved than the next one, but you can still refer to its source code in `sbcolumn.js`.
- `IncludeItselfOption` is responsible for the "Include itself" checkbox and "owns" the `includeItself` specification parameter. This is one of the simplest options imaginable, so we'll look at its code in detail.

sbcolumn.js

```
var IncludeItselfOption = api.subClass('IncludeItselfOption', api.ColumnOption, {
  createInput: function(div$) {
    this.checkbox$ = div$.append(
        AJS.template('<div class="checkbox"><label><input type="checkbox"> 
{label}</label></div>')
          .fill({ label: AJS.I18n.getText("sbcolumn.include-itself") })
          .toString()).find('input');
    var params = this.spec.params;
    this.checkbox$.on('change', function() {
      if ($(this).is(':checked')) {
        params.includeItself = true;
      } else {
        delete params.includeItself;
      }
      div$.trigger('notify');
    });
  },
  notify: function() {
    this.checkbox$.prop('checked', !!this.spec.params.includeItself);
    return true;
  }
});
```

Because the option class specifies no `title` and doesn't override `createLabel()`, there is no label to the left of the checkbox.

The `createInput()` method creates the checkbox and sets up event handling. It is passed a jQuery object to append the input elements to.

Please note that Structure column configuration panels use the AUI Forms[687] HTML layout (with modified CSS styles). You should use the same layout in your HTML code to make your options look consistent with Structure's. In the example above, the checkbox is wrapped in a `<div class="checkbox">` element to comply with AUI Forms.

Also note how the `change` event handler of the checkbox modifies the current specification parameters and always triggers a `notify` event on the provided jQuery object. These are the crucial parts of the option contract.

The `notify()` method is called whenever the current specification changes. Its job is to transfer the data in the opposite direction – from the specification to the input elements. This method also decides whether the option is applicable – if it returns a "falsy" value, the option's row on the configuration panel is hidden from the user.

687 https://developer.atlassian.com/display/AUI/Forms

`ColumnType`

The `api.ColumnType`(see page 1144) class is the main entry point used by the Structure plugin to call your client-side column code. A column type instance creates column presets, columns, and configurators. To find the complete source code for the Status Bar column type, please open `sbcolumn.js` from the API example sources(see page 1152) in your favorite editor and scroll to the `StatusBarType` class definition.

The `getMetadataRequests()` method declares the column-level metadata request to load the available issue statuses from JIRA. See Requesting and Using Metadata(see page 1057) above for details.

The `createSwitchTypePreset()` method creates a single column specification, which is used as a preset when the user selects our type in the "Type" drop-down on the column configuration panel.

Note the call to the `isAvailable()` function that checks that the preset is needed for the primary panel and that the status metadata is indeed available. If that check fails, the method returns `null`, making it impossible to switch to the Status Bar column type. You can try it yourself – open the Search Result secondary panel, add any column to it and try to change its column type. You should see that the Status Bar type is not available.

The switching preset doesn't have to be fully configured, because the configuration panel is already open when it's used. However, because the Status Bar column configuration is quite complex, we make an extra effort and pre-populate the preset with all the known statuses and some default colors for them. This way the user will quickly see what a status bar looks like without having to configure anything at all. This tactic can be useful for other columns with a lot of parameters.

The `createAddColumnPresets()` method creates an array of column specifications that will be used as presets in the "Add Column" menu. Unlike the "switch" preset above, these presets must be completely configured. Like `createSwitchTypePreset()`, this method calls `isAvailable()` first, so a Status Bar column cannot be added to a secondary Structure panel.

Because the "Add Column" menu is the first place where the user discovers your column type, it would be best if your presets are interesting and cover the whole range of the type's functionality. It's not easy to be creative with the Status Bar column though, unless we know the semantics of statuses, which can be arbitrary. So, for simplicity `StatusBarType` adds only a single preset to the "Add Column" menu, reusing the "switch" preset, which is fully configured.

Besides the usual `key`, `name`, and `params`, the "add" presets can have two special properties:

- `presetName` is a string that specifies the name of the preset in the "Add Column" menu. This name will be used *only in the menu*, the added column will have either the `name` from the specification or the default name generated for it. If omitted, the column name will be used as the preset name.
- `shouldOpenConfigurator` – if this flag is set to `true`, the column configuration panel will open immediately after adding the column with this preset. This can be used to create a "Custom..." kind of preset that lets the user explore the available options.

The `createColumn()` and `createConfigurator()` methods return a `Column` or a `ColumnConfigurator` for the given specification, respectively. The methods are similar – they check whether the type is available and the given specification is valid, and if both checks succeed, they instantiate the appropriate subclass. Please note how the column context and the specification are passed to the constructors, this is crucial.

Finally, at the end of the script we instantiate and register our column type, making it available to Structure:

sbcolumn.js

```
api.registerColumnType(new StatusBarType(), COLUMN_KEY);
```

Structure will use our column type instance to handle the columns with the given key. You can also pass an array of keys as the second argument, to associate your type with more than one column key.

Column Groups

Column groups are used to organize column presets in the "Add Column" menu. Each group has a string key and a human-readable name. Column configurator's `getGroupKey()` method should return the appropriate group key for its preset specification.

Structure specifies four column groups for its built-in columns – `fields`, `icons`, `totals`, and `progress`. For the Status Bar column we will register a separate column group:

sbcolumn.js

```
api.registerColumnGroup({ groupKey: GROUP_KEY, title: AJS.I18n.getText("sbcolumn.name
"), order: 1000 });
```

The `order` parameter determines the position of the group within the menu. The higher the order, the lower the group will be. Structure's predefined groups have order between 100 and 400, inclusive.

Web Resources and Contexts

You need to register your JavaScript and CSS code as a web resource in the plugin descriptor. The Status Bar column has no CSS of its own, and all of its JavaScript code is in a single file, `sbcolumn.js`. Because we use the Structure JavaScript API and the `AJS.template()` function from the Atlassian API, we need to declare two dependencies. We also declare a resource transformation to make `AJS.I18n.getText()` calls work.

atlassian-plugin.xml

```
<web-resource key="wr-sbcolumn" name="web-resource:Status Bar Column">
  <dependency>com.atlassian.auiplugin:ajs</dependency>
  <dependency>com.almworks.jira.structure:widget</dependency>
  <transformation extension="js">
    <transformer key="jsI18n"/>
  </transformation>
  <resource type="download" name="sbcolumn.js" location="js/sbcolumn/sbcolumn.js"/>
  <context>structure.widget</context>
</web-resource>
```

We use `structure.widget` web resource context to make our JavaScript (and CSS, if we had any) load on Structure Board. It also works for the Structure's Dashboard Gadget. However, if you'd like your column to work on other pages – Project page, Issue page or Agile Board page, you need to include other web contexts too – see Loading Additional Web Resources For Structure Widget(see page 1065).

10.4.1.7 Export Renderers

Any structure can be exported into printable HTML and Microsoft Excel formats. Exporting is different from rendering the Structure widget in several aspects:

- It is entirely a server-side task, so the code is written in Java.

- The data needed for exporting need not be transferred over the network and cached.
- The export result need not be updated as the exported issues or structure change.
- There are two distinct formats, or media, that are quite different from each other. More formats may be added in the future.

It is because of these differences, that the exporting architecture and APIs are different from their widget rendering counterparts, being simpler in some aspects and more complex in others, while quite similar overall, sometimes making it non-trivial to avoid "repeating yourself".

Please refer to the javadocs[688] for an overview of the export API and SPI. In short, to export a column, you need to write and register an **export renderer provider**, that would recognize the column specification and return an **export renderer** instance for the given column and export format. The returned renderer will then be given an **export column** instance to configure and **export cell** instances to render the values. The **export context** and **export row** instances will provide all the data, including the required attributes.

Speaking of the interfaces that must be implemented, `ExportRendererProvider`[689] is analogous to `AttributeLoaderProvider`, and `ExportRenderer`[690] is a mixture of `AttributeLoader` and the client-side `Column`(see page 1135).

Export Strategies

The main difficulty with export is having different output formats with different features. For example, if you have a method for converting a value to HTML, you could reuse it for the printable HTML export. But when exporting to Excel, HTML support is very limited, and if your values correspond to one of Excel's data types, e.g. date, you need to set an appropriate column style. On the other hand, if you have a simple plain-text column, the format doesn't matter – you can have a single export renderer that calls `setText()` on any type of cell.

The export SPI is flexible, and allows you to use different strategies for different column types. There are three basic kinds of export renderer providers.

- A **specific renderer provider** declares which particular export format it supports in the plugin descriptor. It is parameterized with the expected column and cell types, and returns similarly parameterized renderers, that use format-specific methods.
- A **generic renderer provider** does not declare an export format in the plugin descriptor, so its priority is lower than that of a specific renderer provider. It returns generic renderers, that only call the methods of the basic `ExportCell` and `ExportColumn` interfaces. Though limited, such a provider will work for any other export format that may be added in the future.
- A **multi-format renderer provider** either declares no supported formats (like a generic provider), or declares multiple supported formats. It is not parameterized with specific cell and column types, but it keeps track of the current export format, and its renderers may call format-specific methods by casting the given column and cell instances to the appropriate types. Though more difficult to write, a multi-format provider can combine the benefits of generic and specific providers and help avoid code duplication.

Exploring the extremes, we will create two export renderer providers for the Status Bar column. The first will be a generic provider, that will present the data as plain text instead of drawing a progress bar. The second one will be an advanced Excel provider that will use the underlying low-level Apache POI API to draw pseudo-graphic progress bars in Excel cells.

688 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/export/package-summary.html
689 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/export/ExportRendererProvider.html
690 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/export/ExportRenderer.html

Generic Renderer Provider

The `StatusBarRendererProvider` class in the `status-bar-column` example plugin source contains both the generic provider and its renderer. The code is quite long, but that's mostly due to defensive checks and the general verbosity of Java. The operation of both the provider and the renderer is quite straight-forward.

The provider's `getColumnRenderer()` method does the following:

- Checks that the given column specification indeed represents a Status Bar column, just in case.
- Obtains the column name form the specification, generating a default name if there is none.
- Extracts the `statuses` array and the `includeItself` flag from the specification parameters. These are needed for rendering.
- Creates and returns an instance of the `StatusBarRenderer` inner class, passing it the column name and parameters.

The renderer has `prepare()` method that lets it specify which attributes it will need loaded to do the export. Like in `StatusBarColumn`, we request our histogram-based custom attribute and status for the current row.

The renderer's `configureColumn()` method sets the column name by calling `setText()` on the given column's header cell.

The renderer's `renderCell()` method does the following:

- Obtains the attribute values from the context.
- Adjusts the data if the the "Include itself" option is off, by decrementing the issue count for the current issue's status.
- Iterates over the selected statuses, adding each non-zero sub-issue count and the corresponding status name to a `StringBuilder`.
- If the resulting value is not empty, calls `setText()` on the given cell.

Here is the module declaration for the generic renderer provider. Note that it specifies the column key, but no export format.

atlassian-plugin.xml

```
<structure-export-renderer-provider key="erp-sbcolumn" name="export-renderer:Status
Bar Column Provider"
                                    class="com.almworks.jira.structure.sbcolumn.Statu
sBarRendererProvider">
  <column-key>com.almworks.jira.structure.sbcolumn</column-key>
</structure-export-renderer-provider>
```

Advanced Excel Renderer Provider

The `StatusBarExcelProvider` class contains the advanced Excel renderer and the corresponding provider.

The provider's `getColumnRenderer()` method is very similar to the generic provider's, with two additions:

- it checks that the export format is indeed `MS_EXCEL`;
- it also extracts the `colors` array from the specification parameters, as the renderer will use those (or similar) colors for the progress bar.

The renderer's `prepare()` and `configureColumn()` methods are the same as the generic version. The `renderCell()` method begins in a similar way, by extracting the data map and adjusting it for the "Include itself" option, if needed.

The interesting part is the actual rendering. The pseudo-graphic "progress bar" that the renderer creates is a string of 30 "pipe" characters, split into colored stripes with lengths proportional to issue counts. `ExcelCell` provides no support for rich text formatting (besides `setRichTextFromHtml()`, which is not up to the task), but we can access the lower-level API, Apache POI HSSF[691], by obtaining the underlying POI objects from `ColumnContext.getObject()`[692] using the keys from `ColumnContextKeys.Excel`[693].

The code that distributes the 30 characters among the stripes is ported from `sbcolumn.js`. To completely understand how the rich text part works, you'll need some knowledge of the POI HSSF API, which is quite complex and outside of the scope of this document. Please refer to the POI documentation and the `StatusBarExcelProvider` source code for more information.

The module declaration for the Excel renderer provider is given below. Note that it specifies both a column key and an export format, thus overriding the generic provider for the Excel format.

atlassian-plugin.xml

```
<structure-export-renderer-provider key="erp-sbcolumn-excel" name="export-
renderer:Status Bar Column Excel Provider"
                                    class="com.almworks.jira.structure.sbcolumn.Statu
sBarExcelProvider">
  <column-key>com.almworks.jira.structure.sbcolumn</column-key>
  <export-format>ms-excel</export-format>
</structure-export-renderer-provider>
```

10.4.2 Loading Additional Web Resources For Structure Widget

To include a web resource (such as custom CSS or JavaScript file) on the page every time Structure Widget(see page 88) is displayed, use `structure.widget` web resource context and possibly a few others.Use cases:

1. You create your own custom field and would like it to be editable in the Structure grid. The field is powered by additional JavaScript or CSS, which should be loaded on the page that displays structure.
2. You create your own column type(see page 1048). You'll need to use the Structure JavaScript API(see page 1132) and register the web resource with your JavaScript code as a widget extension.

10.4.2.1 Using Web Resource Contexts

You can add JavaScript or CSS to the Structure widget by adding a web resource to the `structure.widget` context. Note, however, that due to Atlassian API limitations, context-provided web resources may not be loaded on **all** pages with the Structure widget. The following table lists all web resource contexts related to pages where Structure Widget can possibly be shown.

Web Resource Context	Used on...
`structure.widget`	Structure Board, Structure Gadget
`jira.view.issue`	Issue Page, Issue Navigator in details view

691 http://poi.apache.org/spreadsheet/index.html
692 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/column/ColumnContext.html#getObject(java.lang.Object)
693 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/column/ColumnContextKeys.Excel.html

Web Resource Context	Used on...
`gh-rapid`	Scrum and Kanban boards from JIRA Software
`jira.browse.project`	Project page (including Structure tab)
`structure.printable`	Printable Structure page

To have your code present on every page where a Structure widget can possibly be shown, include all these resources. You usually don't need to include `structure.printable` though, unless you have some special rules for printing.

Sample snippet from `atlassian-plugin.xml`:

```
<web-resource key="custom-field-resource" name="My Custom Field Web Resource">
  <resource type="download" name="custom-field-resource.js" location="js/myplugin/
custom-field-resource.js"/>
  <context>structure.widget</context>
  <context>jira.view.issue</context>
  <context>gh-rapid</context>
  <context>jira.browse.project</context>
</web-resource>
```

10.4.3 Declaring a New Generic Item Type

Generic items are managed by Structure and are similar to folders but may also contain an icon and a description. You can define a generic item type in your app to allow Structure users to create and work with domain-specific items, e.g. milestones or test cases. A new generic item type is defined in your app's `atlassian-plugin.xml` by declaring a `<structure-item-type>` module
and using `com.almworks.jira.structure.api.item.generic.GenericItemType` as its implementation class.

10.4.3.1 Example

```
<structure-item-type key="type-milestone" name="itemtype:Milestone" weight="100"
                     class="com.almworks.jira.structure.api.item.generic.GenericItemT
ype">
 <icon spanClass="my-app-milestone-icon-class"/>
 <displayable key="my.app.milestone.displayable"/>
 <title key="my.app.milestone.title"/>
 <newItemTitle key="my.app.milestone.new"/>
</structure-item-type>
```

Element	Required	Description
`@key`	Yes	Unique module key within the plugin. Full module key will define the `itemType` part of `ItemIdentity`[694].
`@name`	No	A human-readable name of the plugin module.
`@weight`	No	Determines the order in which generic item types appear in menus and lists. Items with the 'lightest' weight are displayed first and the 'heaviest' items sink to the bottom.
`@class`	Yes	Module class. Must be `GenericItemType`[695].
`icon`	No	An icon shown in the item row. At this point the only supported option is using a single icon, associated with a CSS class, for all items of the given type. If you're using your own icons, make sure the appropriate CSS styles are loaded everywhere Structure can be used (see Loading Additional Web Resources For Structure Widget[696]).
`displayable`	Yes	An I18N key used to generate a textual representation of an item for activity streams, decision panels, and elsewhere. The value should contain the type name and a placeholder for the item name, e.g. "milestone {0}".
`title`	Yes	An I18N key for the item creation panel, e.g. "Milestone".
`newItemTitle`	Yes	An I18N key for the Structure toolbar's "Add item" drop-down menu, e.g. "New Milestone".

10.4.3.2 Programmatic Access to Generic Items

Use `GenericItemService`[697] or `GenericItemManager`[698] to create, retrieve, and update generic items in your plugin code. `GenericItemService` is a higher-level component which checks users' permissions and performs other validation tasks as needed. `GenericItemManager` is a low-level component which queries and updates the database, throwing exceptions if anything goes wrong.

694 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/ItemIdentity.html
695 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/generic/GenericItemType.html
696 https://wiki.almworks.com/display/structure2gmaster/Loading+Additional+Web+Resources+For+Structure+Widget
697 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/generic/GenericItemService.html
698 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/generic/GenericItemManager.html

10.4.3.3 Generic Item Permissions

Each generic item is associated with the structure that contains it, and that structure's permissions are used to determine who can see and update the item.

- Any user can create a new generic item programmatically. **Edit** access level is required to add the new item to a structure. When the item is added to a structure, it becomes associated with that structure.
- As with issues and folders, **Edit** access level is required to create a generic item using Structure UI. The item is associated with the structure it was created in.
- If a generic item from one structure is copied or moved to a different structure, a copy of the item is created and associated with the new structure.
- All users having the **View** access level to a structure can view all generic items in that structure.
- All users having the **Edit** access level to a structure can update and delete all generic items in that structure.
- As with any other item, **Edit** access level is required to remove a generic item from a structure. When a generic item is removed from a structure it is **not** deleted from the database. It can still be seen in structure history(see page 556), accessed or updated programmatically, and re-inserted into the structure.

10.5 Accessing Structure Data Remotely

Structure plugin provides REST API, which is primarily used by the structure widget(see page 84). The same API can be used to access the hierarchical data remotely from an automation script or another user agent application.

See details in the REST API Reference(see page 1086).

10.6 Reference

10.6.1 Structure Developer Reference

10.6.2 Structure Java API Reference

> Structure API is work in progress. You will find that some of the packages are documented less than others, and some are not documented yet.
>
> We're constantly working on the API improvements and documentation and will make the javadocs and other parts of the documentation more complete with every release.

Structure API Reference for the latest version: http://almworks.com/structure/javadoc/latest

You can download javadocs from the Maven repositories into your IDE.

Check out information about Structure API Versions(see page 1069) to select the correct API artifact, and you can also download Javadoc JARs there.

10.6.2.1 Structure API Versions

Current Versions

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
17.18.0 Javadocs[699]	Jira 8.20+	8.3.0+	"[17.18,18)"	2023-04-25
17.17.0 Javadocs[700]	Jira 8.13+	8.2.1+	"[17.17,18)"	2023-01-09

699 https://almworks.com/structure/javadoc/17.18.0/
700 https://almworks.com/structure/javadoc/17.17.0/

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
17.16.0 Javadocs[701]	Jira 8.13+	8.2.0+	"[17.16,18)"	2022-12-15
17.15.0 Javadocs[702]	Jira 8.13+	8.1.0+	"[17.15,18)"	2022-08-22
17.13.0 Javadocs[703]	Jira 8.13+	8.0.0+	"[17.13,18)"	2022-03-30
17.12.0 Javadocs[704]	Jira 8.13+	7.4.0+	"[17.12,18)"	2021-12-21
17.11.0 Javadocs[705]	Jira 8.5+	7.3.0+	"[17.11,18)"	2021-10-22
17.10.0 Javadocs[706]	Jira 8.5+	7.1.0+	"[17.10,18)"	2021-07-22
17.9.0 Javadocs[707]	Jira 8.5+	7.0.0+	"[17.9,18)"	2021-06-04
17.8.0 Javadocs[708]	Jira 8.5+	6.6.0+	"[17.8,18)"	2021-03-19

701 https://almworks.com/structure/javadoc/17.16.0/
702 https://almworks.com/structure/javadoc/17.15.0/
703 https://almworks.com/structure/javadoc/17.13.0/
704 https://almworks.com/structure/javadoc/17.12.0/
705 https://almworks.com/structure/javadoc/17.11.0/
706 http://almworks.com/structure/javadoc/17.10.0
707 http://almworks.com/structure/javadoc/17.9.0
708 http://almworks.com/structure/javadoc/17.8.0

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
17.7.0 Javadocs[709]	Jira 8.5+	6.5.0+	"[17.7,18)"	2021-02-01
17.6.0 Javadocs[710]	Jira 7.13+	6.4.0+	"[17.6,18)"	2020-12-01
17.5.0 Javadocs[711]	Jira 7.13+	6.3.2+	"[17.5,18)"	2020-11-18
17.4.0 Javadocs[712]	Jira 7.13+	6.3.0+	"[17.4,18)"	2020-10-08
17.3.0 Javadocs[713]	Jira 7.13+	6.2.0+	"[17.3,18)"	2020-08-10
17.2.0 Javadocs[714]	Jira 7.13+	6.1.0+	"[17.2,18)"	2020-06-03
17.1.0 Javadocs[715]	Jira 7.13+	6.0.0+	"[17.1,18)"	2020-03-20
17.0.0 Javadocs[716]	Jira 7.13+	6.0.0.eap1+	"[17,18)"	2020-01-30

709 http://almworks.com/structure/javadoc/17.7.0
710 http://almworks.com/structure/javadoc/17.6.0
711 http://almworks.com/structure/javadoc/17.5.0
712 http://almworks.com/structure/javadoc/17.4.0
713 http://almworks.com/structure/javadoc/17.3.0
714 http://almworks.com/structure/javadoc/17.2.0
715 http://almworks.com/structure/javadoc/17.1.0
716 http://almworks.com/structure/javadoc/17.0.0

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
16.16.0 Javadocs[717]	Jira 7.6+	5.6.0+	"[16.16,17)"	2019-09-18
16.15.0 Javadocs[718]	Jira 7.6+	5.5.0+	"[16.15,17)"	2019-07-08
16.14.0 Javadocs[719]	Jira 7.6+	5.4.0+	"[16.14,17)"	2019-06-03
16.13.0 Javadocs[720]	Jira 7.6+	5.3.0+	"[16.13,17)"	2019-03-22
16.12.0 Javadocs[721]	Jira 7.6+	5.2.0+	"[16.12,17)"	2018-12-25
16.11.0 Javadocs[722]	Jira 7.2+	5.1.0+	"[16.11,17)"	2018-10-25
16.10.0 Javadocs[723]	Jira 7.2+	5.0.0+	"[16.10,17)"	2018-08-16
16.9.0 Javadocs[724]	Jira 7.2+	4.6.0+	"[16.9,17)"	2018-03-28

717 http://almworks.com/structure/javadoc/16.16.0
718 http://almworks.com/structure/javadoc/16.15.0
719 http://almworks.com/structure/javadoc/16.14.0
720 http://almworks.com/structure/javadoc/16.13.0
721 http://almworks.com/structure/javadoc/16.12.0
722 http://almworks.com/structure/javadoc/16.11.0
723 http://almworks.com/structure/javadoc/16.10.0
724 http://almworks.com/structure/javadoc/16.9.0

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
16.8.0 Javadocs[725]	Jira 7.2+	4.5.0+	"[16.8,17)"	2017-12-26
16.7.0 Javadocs[726]	Jira 7.2+	4.4.0+	"[16.7,17)"	2017-11-29
16.6.0 Javadocs[727]	Jira 7.2+	4.3.0+	"[16.6,17)"	2017-10-16
16.5.0 Javadocs[728]	Jira 7.2+	4.2.0+	"[16.5,17)"	2017-08-25
16.4.0 Javadocs[729]	Jira 7.1+	4.1.0+	"[16.4,17)"	2017-06-19
16.3.0 Javadocs[730]	Jira 7.1+	4.0.0+	"[16.3,17)"	2017-04-26
16.2.0 Javadocs[731]	Jira 7.0+	3.6.0+	"[16.2,17)"	2017-04-03
16.1.0 Javadocs[732]	Jira 7.0+	3.5.0+	"[16.1,17)"	2017-01-26
16.0.0 Javadocs[733]	Jira 7.0+	3.4.0+	"[16,17)"	2016-12-07

725 http://almworks.com/structure/javadoc/16.8.0
726 http://almworks.com/structure/javadoc/16.7.0
727 http://almworks.com/structure/javadoc/16.6.0
728 http://almworks.com/structure/javadoc/16.5.0
729 http://almworks.com/structure/javadoc/16.4.0
730 http://almworks.com/structure/javadoc/16.3.0
731 http://almworks.com/structure/javadoc/16.2.0
732 http://almworks.com/structure/javadoc/16.1.0
733 http://almworks.com/structure/javadoc/16.0.0

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
Structure API version 16.0.0 is the first public API version for Structure 3.x. For older API versions compatible with Structure 2.x, see Previous API Versions[734].				

Javadocs for the Latest Version[735] — Java API documentation for the latest API version.

To see how to include the API in your project dependencies, read about Accessing Structure from Jira Plugin(see page 1031).

Version Compatibility

Versioning of the API artifact follows these generally accepted rules:

- Major version is increased when the client code – your code – might not compile with the new version.
- Minor version is increased when new methods are added to the API (so your code might break if you downgrade to a lower minor version).
- Micro version is changed when there's no impact on the compatibility.

Getting Versions

The API jars can be downloaded from the public Maven repositories. This is the recommended way.

If you can't download API jars from Maven repository for any reason, you can download them from this page and install into your local Maven repository:

```
mvn install:install-file -Dfile=structure-api-17.18.0.jar -DpomFile=structure-
api-17.18.0.pom
```

File	Modified
structure-api-17.18.0-javadoc.jar[736]	Apr 26, 2023 by Julia Atlygina[737]
structure-api-17.18.0-sources.jar[738]	Apr 26, 2023 by Julia Atlygina[739]
structure-api-17.18.0.jar[740]	Apr 26, 2023 by Julia Atlygina[741]
structure-api-17.18.0.pom[742]	Apr 26, 2023 by Julia Atlygina[743]

734 https://wiki.almworks.com/display/structure0211/Structure+API+Versions
735 http://almworks.com/structure/javadoc/latest
736 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.18.0-javadoc.jar?api=v2
737 https://wiki.almworks.com/display/~atlygina
738 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.18.0-sources.jar?api=v2
739 https://wiki.almworks.com/display/~atlygina
740 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.18.0.jar?api=v2
741 https://wiki.almworks.com/display/~atlygina
742 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.18.0.pom?api=v2
743 https://wiki.almworks.com/display/~atlygina

File	Modified
structure-api-16.16.0-javadoc.jar[744]	Apr 26, 2023 by Julia Atlygina[745]
structure-api-16.16.0-sources.jar[746]	Apr 26, 2023 by Julia Atlygina[747]
structure-api-16.16.0.jar[748]	Apr 26, 2023 by Julia Atlygina[749]
structure-api-16.16.0.pom[750]	Apr 26, 2023 by Julia Atlygina[751]
structure-api-17.8.0-javadoc.jar[752]	Apr 26, 2023 by Julia Atlygina[753]
structure-api-17.8.0-sources.jar[754]	Apr 26, 2023 by Julia Atlygina[755]
structure-api-17.8.0.jar[756]	Apr 26, 2023 by Julia Atlygina[757]
structure-api-17.8.0.pom[758]	Apr 26, 2023 by Julia Atlygina[759]
structure-api-17.9.0-javadoc.jar[760]	Apr 26, 2023 by Julia Atlygina[761]
structure-api-17.9.0-sources.jar[762]	Apr 26, 2023 by Julia Atlygina[763]
structure-api-17.9.0.jar[764]	Apr 26, 2023 by Julia Atlygina[765]
structure-api-17.9.0.pom[766]	Apr 26, 2023 by Julia Atlygina[767]
structure-api-17.10.0-javadoc.jar[768]	Apr 26, 2023 by Julia Atlygina[769]

744 https://wiki.almworks.com/download/attachments/173313598/structure-api-16.16.0-javadoc.jar?api=v2
745 https://wiki.almworks.com/display/~atlygina
746 https://wiki.almworks.com/download/attachments/173313598/structure-api-16.16.0-sources.jar?api=v2
747 https://wiki.almworks.com/display/~atlygina
748 https://wiki.almworks.com/download/attachments/173313598/structure-api-16.16.0.jar?api=v2
749 https://wiki.almworks.com/display/~atlygina
750 https://wiki.almworks.com/download/attachments/173313598/structure-api-16.16.0.pom?api=v2
751 https://wiki.almworks.com/display/~atlygina
752 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.8.0-javadoc.jar?api=v2
753 https://wiki.almworks.com/display/~atlygina
754 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.8.0-sources.jar?api=v2
755 https://wiki.almworks.com/display/~atlygina
756 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.8.0.jar?api=v2
757 https://wiki.almworks.com/display/~atlygina
758 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.8.0.pom?api=v2
759 https://wiki.almworks.com/display/~atlygina
760 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.9.0-javadoc.jar?api=v2
761 https://wiki.almworks.com/display/~atlygina
762 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.9.0-sources.jar?api=v2
763 https://wiki.almworks.com/display/~atlygina
764 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.9.0.jar?api=v2
765 https://wiki.almworks.com/display/~atlygina
766 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.9.0.pom?api=v2
767 https://wiki.almworks.com/display/~atlygina
768 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.10.0-javadoc.jar?api=v2
769 https://wiki.almworks.com/display/~atlygina

File	Modified
structure-api-17.10.0-sources.jar[770]	Apr 26, 2023 by Julia Atlygina[771]
structure-api-17.10.0.jar[772]	Apr 26, 2023 by Julia Atlygina[773]
structure-api-17.10.0.pom[774]	Apr 26, 2023 by Julia Atlygina[775]
structure-api-17.11.0-javadoc.jar[776]	Apr 26, 2023 by Julia Atlygina[777]
structure-api-17.11.0-sources.jar[778]	Apr 26, 2023 by Julia Atlygina[779]
structure-api-17.11.0.jar[780]	Apr 26, 2023 by Julia Atlygina[781]
structure-api-17.11.0.pom[782]	Apr 26, 2023 by Julia Atlygina[783]
structure-api-17.12.0-javadoc.jar[784]	Apr 26, 2023 by Julia Atlygina[785]
structure-api-17.12.0-sources.jar[786]	Apr 26, 2023 by Julia Atlygina[787]
structure-api-17.12.0.jar[788]	Apr 26, 2023 by Julia Atlygina[789]
structure-api-17.12.0.pom[790]	Apr 26, 2023 by Julia Atlygina[791]
structure-api-17.13.0.pom[792]	Apr 26, 2023 by Julia Atlygina[793]
structure-api-17.13.0.jar[794]	Apr 26, 2023 by Julia Atlygina[795]

770 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.10.0-sources.jar?api=v2
771 https://wiki.almworks.com/display/~atlygina
772 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.10.0.jar?api=v2
773 https://wiki.almworks.com/display/~atlygina
774 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.10.0.pom?api=v2
775 https://wiki.almworks.com/display/~atlygina
776 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.11.0-javadoc.jar?api=v2
777 https://wiki.almworks.com/display/~atlygina
778 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.11.0-sources.jar?api=v2
779 https://wiki.almworks.com/display/~atlygina
780 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.11.0.jar?api=v2
781 https://wiki.almworks.com/display/~atlygina
782 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.11.0.pom?api=v2
783 https://wiki.almworks.com/display/~atlygina
784 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.12.0-javadoc.jar?api=v2
785 https://wiki.almworks.com/display/~atlygina
786 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.12.0-sources.jar?api=v2
787 https://wiki.almworks.com/display/~atlygina
788 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.12.0.jar?api=v2
789 https://wiki.almworks.com/display/~atlygina
790 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.12.0.pom?api=v2
791 https://wiki.almworks.com/display/~atlygina
792 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.13.0.pom?api=v2
793 https://wiki.almworks.com/display/~atlygina
794 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.13.0.jar?api=v2
795 https://wiki.almworks.com/display/~atlygina

File	Modified
structure-api-17.13.0-sources.jar[796]	Apr 26, 2023 by Julia Atlygina[797]
structure-api-17.13.0-javadoc.jar[798]	Apr 26, 2023 by Julia Atlygina[799]
structure-api-17.15.0.pom[800]	Apr 26, 2023 by Julia Atlygina[801]
structure-api-17.15.0.jar[802]	Apr 26, 2023 by Julia Atlygina[803]
structure-api-17.15.0-sources.jar[804]	Apr 26, 2023 by Julia Atlygina[805]
structure-api-17.15.0-javadoc.jar[806]	Apr 26, 2023 by Julia Atlygina[807]
structure-api-17.16.0.pom[808]	Apr 26, 2023 by Julia Atlygina[809]
structure-api-17.16.0.jar[810]	Apr 26, 2023 by Julia Atlygina[811]
structure-api-17.16.0-sources.jar[812]	Apr 26, 2023 by Julia Atlygina[813]
structure-api-17.16.0-javadoc.jar[814]	Apr 26, 2023 by Julia Atlygina[815]
structure-api-17.17.0.pom[816]	Apr 26, 2023 by Julia Atlygina[817]
structure-api-17.17.0.jar[818]	Apr 26, 2023 by Julia Atlygina[819]
structure-api-17.17.0-sources.jar[820]	Apr 26, 2023 by Julia Atlygina[821]

796 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.13.0-sources.jar?api=v2
797 https://wiki.almworks.com/display/~atlygina
798 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.13.0-javadoc.jar?api=v2
799 https://wiki.almworks.com/display/~atlygina
800 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.15.0.pom?api=v2
801 https://wiki.almworks.com/display/~atlygina
802 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.15.0.jar?api=v2
803 https://wiki.almworks.com/display/~atlygina
804 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.15.0-sources.jar?api=v2
805 https://wiki.almworks.com/display/~atlygina
806 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.15.0-javadoc.jar?api=v2
807 https://wiki.almworks.com/display/~atlygina
808 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.16.0.pom?api=v2
809 https://wiki.almworks.com/display/~atlygina
810 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.16.0.jar?api=v2
811 https://wiki.almworks.com/display/~atlygina
812 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.16.0-sources.jar?api=v2
813 https://wiki.almworks.com/display/~atlygina
814 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.16.0-javadoc.jar?api=v2
815 https://wiki.almworks.com/display/~atlygina
816 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.17.0.pom?api=v2
817 https://wiki.almworks.com/display/~atlygina
818 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.17.0.jar?api=v2
819 https://wiki.almworks.com/display/~atlygina
820 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.17.0-sources.jar?api=v2
821 https://wiki.almworks.com/display/~atlygina

File	Modified
structure-api-17.17.0-javadoc.jar[822]	Apr 26, 2023 by Julia Atlygina[823]

10.6.3 Structure Plugin Module Types

The following module types are added by the Structure plugin:

- structure-attribute-loader-provider(see page 1078) lets you provide the data for new column types in the Structure widget.
- structure-export-renderer-provider(see page 1079) lets you export new column types to printable HTML and Excel files.
- structure-item-type(see page 1080) lets you define a new type of items, which can be used in structures.
- Generator Modules(see page 1081) let you add generators to the Automation subsystem. The following module types are included:
  - `structure-inserter`
  - `structure-extender`
  - `structure-filter`
  - `structure-grouper`
  - `structure-sorter`
- structure-effector(see page 1082) lets you add effectors to the Automation subsystem.
- structure-effect-provider(see page 1083) lets you define a new effect provider, which can be used by effectors.
- new-structure-template(see page 1085) lets you add templates for new structures.
- structure-query-constraint(see page 1084) allows adding new functions to S-JQL language.
- structure-synchronizer(see page 1161) defines a new synchronizer.

10.6.3.1 structure-attribute-loader-provider

You can use this module to add your support for attributes, either new or already existing, to Structure. The attributes are used by Structure Widget columns, by exporters and by generators.

Example

```
<structure-attribute-loader-provider key="provider-key"
    class="com.company.your.plugin.attribute.MyAttributeProvider"/>
```

Element	Required?	Description
`structure-attribute-loader-provider`	Yes	The module descriptor.
`@key`	Yes	The unique identifier of the plugin module.

822 https://wiki.almworks.com/download/attachments/173313598/structure-api-17.17.0-javadoc.jar?api=v2
823 https://wiki.almworks.com/display/~atlygina

Element	Required?	Description
`@name`	No	The human-readable name of the plugin module.
`@order`	No	Takes an integer number (including negative numbers) that will define the order in which attribute specs are offered to the providers.
`@class`	Yes	The class that implements the data provider. Must implement AttributeLoaderProvider[824].

10.6.3.2 structure-export-renderer-provider

Export renderer provider module lets you register the components responsible for exporting Structure columns to printable HTML and Microsoft Excel formats.

Export renderer provider example

```
<structure-export-renderer-provider
     key="erp-sbcolumn-excel"
     name="export-renderer:Status Bar Column Excel Provider"
     class="com.almworks.jira.structure.sbcolumn.StatusBarExcelProvider">

  <column-key>com.almworks.jira.structure.sbcolumn</column-key>
  <export-format>ms-excel</export-format>
</structure-export-renderer-provider>
```

Element	Required?	Description
`structure-export-renderer-provider`	Yes	The module descriptor.
`@key`	Yes	The unique identifier of the plugin module.
`@name`	No	The human-readable name of the plugin module.

824 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/loader/AttributeLoaderProvider.html

Element	Required?	Description
`@class`	Yes	The class that implements the renderer provider. Must implement ExportRendererProvider[825].
`column-key`	No	The column key that this provider is associated with. You can have multiple `column-key` elements in a single descriptor. If no column key is specified, the renderer provider is considered generic – such a provider will be consulted for every column not served by a type-specific provider.
`export-format`	No	The export format that this provider is associated with. The values are `printable` for the printable HTML format and `ms-excel` for the Microsoft Excel XLS format. You can have multiple `export-format` elements in a single descriptor. If no export format is specified, the renderer provider is considered generic – such a provider will be consulted for every column not served by a format-specific provider.

10.6.3.3 structure-item-type

This module type lets you declare a new item type. Items of that type can then be used in structures.

Example

```
<structure-item-type key="type-book" name="itemtype:Book"
    class="com.mycompany.structure.books.BookItemType"/>
```

Element	Required?	Description
`@key`	Yes	The unique identifier of the plugin module. Full module key will define the `itemType` part of ItemIdentity[826].
`@name`	No	The human-readable name of the plugin module.

825 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/export/ExportRendererProvider.html
826 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/ItemIdentity.html

Element	Required?	Description
@class	Yes	The class that implements the support for the item type. Must implement StructureItemType[827].

10.6.3.4 Generator Modules

There are five modules, one for each type of generators, that work in the same way:

- structure-inserter
- structure-extender
- structure-filter
- structure-grouper
- structure-sorter

Each module allows declaring a generator of a specific type. When a plugin with a generator module is installed, you get the ability add those generators to structures.

Example

```
<structure-extender
    key="extender-examples" name="extender:Examples" description="Examples extender"
    class="com.mycompany.structure.examples.ExamplesExtender">
  <label key="com.mycompany.examples.extender.label"/>
  <icon spanClass="s-fa s-fa-link"/>
  <dialog-title key="com.mycompany.examples.extender.dialog-title"/>
  <resource type="velocity" name="form" location="/templates/example/extender-
examples.vm"/>
  <resource type="velocity" name="summary" location="/templates/example/extender-
examples-summary.vm"/>
</structure-extender>
```

Other types of generators are declared in the same way.

Element	Required?	Description
@key	Yes	The unique identifier of the generator. Full module key will a part of generator specification, defining the automation.
@name	No	The name of the module for the JIRA administrator.
@description	No	Description of the module for the JIRA administrator.

827 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/StructureItemType.html

Element	Required?	Description
`label`	Yes	The name of the generator as it appears to the user.
`icon`	No	The icon for the generator that will be shown whenever the generator row is displayed. See below for details.
`dialog-title`	No	The title of the dialog that is used to edit the generator
`resource[@name=form]`	No	Form template that will be used for editing the generator's parameters
`resource[@name=summary]`	No	Form template that will be used to display the generator as a row in a structure

Generator Icons

The icon for the generator is defined using CSS classes. If you're using your own icons, make sure the appropriate CSS styles are loaded everywhere Structure can be used (see Loading Additional Web Resources For Structure Widget(see page 1065)).

You can also use the standard icons used by bundled generators:

Generator Type	Icon Classes
Inserter	`s-fa s-fa-plus`
Extender	`s-fa s-fa-link`
Filter	`alm alm-group`
Grouper	`s-fa s-fa-filter`
Sorter	`alm alm-sort-asc`

10.6.3.5 structure-effector

You can use this module to add a new effector to Structure. Effectors are used to update items based on Structure forests and attribute values.

Example

```
<structure-effector
    key="effector-example" name="effector:Example"
    class="com.mycompany.structure.examples.ExamplesEffector">
  <label key="com.mycompany.examples.effector.label"/>
  <dialog-title key="com.mycompany.examples.effector.dialog-title"/>
  <resource type="velocity" name="form" location="/templates/example/effector-
example-form.vm"/>
  <resource type="velocity" name="summary" location="/templates/example/effector-
example-summary.vm"/>
</structure-extender>
```

Element	Required?	Description
`@key`	Yes	The unique identifier of the effector. Full module key will be a part of effector specification, defining the automation.
`@name`	No	The name of the module for the Jira administrator.
`@description`	No	Description of the module for the Jira administrator.
`@class`	Yes	Fully-qualified class name of the effector implementation. Must implement Effector[828].
`label`	Yes	The name of the effector as it appears to the user.
`dialog-title`	No	The title of the dialog that is used to edit the effector.
`resource[@name=form]`	No	Velocity template with the HTML form that will be used for editing the effector's parameters.
`resource[@name=summary]`	No	Velocity template that will be used to display the effector as a row in a structure.

10.6.3.6 structure-effect-provider

You can use this module to add your new effect provider to Structure. Effect providers are used by effectors, e.g. to update issue field values.

828 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/Effector.html

Example

```
<structure-effect-provider key="provider-key"
    class="com.company.your.plugin.MyEffectProvider"/>
```

Element	Required?	Description
`structure-effect-provider`	Yes	The module descriptor.
`@key`	Yes	The unique identifier of the plugin module.
`@name`	No	The human-readable name of the plugin module.
`@class`	Yes	The class that implements the effect provider. Must implement EffectProvider[829].

10.6.3.7 structure-query-constraint

Structure Query Constraint module allows you to define an additional constraint function that can be used in S-JQL.

For example, `folder()` function explained in S-JQL Reference(see page 434) is implemented with a `structure-query-constraint`.

Example:

```
<structure-query-constraint key="constraint-foo"
                            class="com.mycompany.structure.FooConstraint"
                            name="Structure Query Constraint: foo"
                            fname="foo"/>
```

Element	Required?	Description
`key`	Yes	Module key.
`name`	No	Module name for the JIRA administrator.
`class`	Yes	Class that implements StructureQueryConstraint[830].

829 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/EffectProvider.html
830 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/query/StructureQueryConstraint.html

Element	Required?	Description
fname	Yes	Function name, must be unique throughout the system.

10.6.3.8 new-structure-template

New Structure Template module allows you to add templates to the Create Structure dialog.

Example:

```
<new-structure-template key="big-template"
    class="com.mycompany.structure.template.bigtemplate"
    name="New Structure Template: Big Template">
  <label key="com.mycompany.template.big-template.label"/>
  <description key="com.mycompany.template.big-template.description"/>
  <resource type="download" name="icon.png" location="css/structure/templates/
big@2x.png"/>
  <resource type="velocity" name="step1" location="templates/structure/big/step1.vm"/
>
  <resource type="velocity" name="step2" location="templates/structure/big/step2.vm"/
>
</new-structure-template>
```

Element	Required?	Description
@key	Yes	Module key.
@name	No	The name of the module for JIRA administrators.
@class	Yes	The class that implements the template, must implement NewStructureTemplate[831].
label	Yes	The name of the template as it appears in the Create Structure dialog.
description	No	Description of the template.
resource[@type=velocity]	No	Any number of HTML templates used by your code to render wizard steps.
resource[@type=download]	No	Any number of downloadable images or other resources used by your template.

831 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/template/NewStructureTemplate.html

10.6.4 Structure REST API Reference

Structure REST API is under development. The functionality available through REST is sometimes not complete, but it allows to work with the structures.

API version 2 is also not stable, although we're not seeing major changes coming to the main resources.

Both version 1 and version 2 of the REST APIs have been driven by the needs of Structure Widget. We're currently developing a higher-level API specifically for integrations rather than for the product itself. Let us know at Tempo Support[832] if you'd like to contribute or get preliminary access to that API.

10.6.4.1 General Notes

API Versions

As of Structure version 3.4, there are two versions of the REST API – `1.0` and `2.0`. Some of the REST resources are exposed through version 1.0 and some through version 2.0.

Version 1.0 is stable and we don't plan to change it. It comes from Structure 2 and largely remains the same as in Structure 2.x versions. Some of the resources may become deprecated as we replace them with the newer versions.

Version 2.0 is not stable and is being developed along with the product. That means that you can use it, but you need to test your integration every time you upgrade. We are also going to publish API changes in the release notes.

REST Resource Addresses

Structure REST API resources have the URL

```
BASEURL/rest/structure/VERSION/NAME
```

where BASEURL is the base JIRA address (`http://localhost:2990/jira` being standard base URL for development environment), VERSION is the version of the API (either 1.0 or 2.0) and NAME is the name of the resource. For each documented resource there's an indication about its API version.

Authentication

Authentication is done via standard JIRA authentication engine and supported by cookies. When accessing REST API from a remote application, you may need to set up the session first by calling JIRA authentication REST resource. (You don't need to do that if you access Structure REST API from a JavaScript on a page from the same JIRA instance.)

Most read operations are available to non-authenticated access (subject to permission checks for the anonymous user). Most mutation operations are available to authenticated users only.

832 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

10.6.4.2 REST Resources

- Structure Resource(see page 1087) is used to create and manage structures (but not the content)
- Forest Resource(see page 1105) is used to retrieve and update forests (a structure's content)
- Item Resource(see page 1108) is used to create and update items (issues, folders and, possibly, items of other types)
- Value Resource(see page 1112) is used to retrieve attribute values for a given forest
- Attribute Subscription Resource(see page 1116) allows subscribing to a particular set of values identified by attributes and rows, and receive updates
- Configuration Resource(see page 1121) lets you manage global permissions and the list of projects for which Structure is enabled

10.6.4.3 Structure Resource

This page describes resources with which you can list(see page 1091), create(see page 1096), read(see page 1099), update(see page 1101), and delete(see page 1104) structures. Structures contain general information(see page 539) such as name and permissions, but not the hierarchy itself. Issue hierarchy is accessed through the Forest Resource(see page 1105). This page also documents structure shape(see page 1087) and its fields(see page 1088), and the error entity(see page 1091) that may be returned in case of the REST API user error.

Structure resource belongs to **version 2.0** of the API.

`/structure/ GET`(see page 1091)	list structures
`/structure/ POST`(see page 1096)	create a structure
`/structure/{id} GET`(see page 1099)	read structure
`/structure/{id}/update POST`(see page 1101)	update one or several structure fields
`/structure/{id} DELETE`(see page 1104)	delete structure

Quick navigation:

- Structure Representations(see page 1087)
- Structure Fields(see page 1088)
- Permission Rules(see page 1089)
- Error Entity(see page 1091)

Structure Representations

Structure is represented via JSON. All resources are also capable of producing XML.

```
{
  "id": 103,
  "name": "Structure with all fields",
  "description": "Voilà! This structure exhibits all fields.",
  "readOnly": "true",
  "editRequiresParentIssuePermission": true,
  "permissions": [
    {
      "rule": "apply",
      "structureId": 102
    },
    {
      "rule": "set",
      "subject": "group",
      "groupId": "jira-developers",
      "level": "edit"
    },
    {
      "rule": "set",
      "subject": "projectRole",
      "projectId": 10010,
      "roleId": 10020,
      "level": "admin"
    },
    {
      "rule": "set",
      "subject": "anyone",
      "level": "view"
    },
    {
      "rule": "set",
      "subject": "user",
      "username": "agentk",
      "level": "none"
    }
  ],
  "owner": "user:admin"
}
```


Top(see page 1087)

Structure Fields

Structure objects accessible through these resources have the following fields, most of which represent structure details as outlined in the Structure User's Guide(see page 539):

`id`	The ID of the structure (integer, `1..2^63 - 1`.)
`name`	The name of the structure. A structure must have a non-empty name, which does not have to be unique.

`description`	The description on the structure. May be absent.
`readOnly`	`true` if the user has only View(see page 543) access level to the structure, otherwise absent.
`editRequiresParentIssuePermission`	`true` if the Require Edit Issue Permission on Parent Issue(see page 545) flag is set on this structure, otherwise absent.
`permissions`	The list of structure permission rules(see page 544). Present only if the user has Control(see page 543) access level to the structure. Some resources do not include permissions unless requested to do so. List order is as important as the rules themselves.
`owner`	The owner(see page 539) of the structure. Present only if the user is the owner of this structure or if he has Browse Users[833] permission. A string of the form `user:USERNAME`, where USERNAME is the JIRA user login name. Example: `user:jsmith`.

Please note that structure resources described on this page do not include information about issue hierarchies. The content of a structure, i.e. its hierarchy of items, can be read or modified using Forest Resource(see page 1105).

Top(see page 1087)

Permission rules

There are two types of permission rules, those that *set* permissions and those that *apply* permissions from another structure. They have different fields depending on the type.

Set rules

`rule`	Must be equal to `set`, case-insensitive.
`subject`	Identifies the type of the subject to which the rule applies. Must be one of `group, projectRole, user, anyone`. See below how to identify the subject.
`level`	Access level to set to the specified subject. Must be equal to one of the names of the Permission Level[834] enum constants, case-insensitive. Please note that Control permission is represented by the ADMIN[835] enumeration constant.

In addition, there are fields to identify the subject.

833 https://confluence.atlassian.com/display/JIRA/Managing+Global+Permissions
834 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html
835 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html#ADMIN

`group`
The rule applies to all users within the JIRA group.

`groupId`	The name of the JIRA group. Example: `jira-developers`.

REST API user can create such rule only for a group he belongs to.

`projectRole`
The rule applies to all users that have a role in a project.

`projectId`	The ID of the project. Example: `10010`.
`roleId`	The ID of the role. Example: `10010`.

REST API user can create such rule only for roles in projects where Structure is enabled, and for which he has Browse Projects[836] *permission.*

`user`
The rule applies to the user.

`username`	Name of the user. Example: `jsmith` for user John Smith.

REST API user can create such rule only if he has Browse Users[837] *permission, and if such user exists.*

`anyone`
The rule applies to all users, even anonymous (not authenticated.) The rule shouldn't have any additional fields.

Apply rules

`rule`	Must be equal to `apply`, case-insensitive.
`structureId`	The ID of the structure which permissions should be applied. Example: `112`

Apply rule creates a dependency on another structure. Circular dependencies are not allowed. Also, a REST API user can create such rule only if he has Control(see page 543) *access level to the referenced structure.*

Top(see page 1087)

836 https://confluence.atlassian.com/display/JIRA/Managing+Project+Permissions
837 https://confluence.atlassian.com/display/JIRA/Managing+Global+Permissions

Error entity

```
{
  "code": 4005,
  "error": "STRUCTURE_NOT_EXISTS_OR_NOT_ACCESSIBLE[4005]",
  "structureId": 160,
  "message": "Referenced structure [160] does not exist or you don't have Control
permissions on it.",
  "localizedMessage": "Das Struktur [160] existiert nicht oder sie haben keine
Kontrolle Berechtigungen."
}
```

In some cases, requests to structure resources result in an error response containing an error entity. Any of its fields may be absent.

`code`	Integer code of the error
`error`	Brief technical description of the error. Contains a name of the corresponding StructureError[838] enum constant.
`structureId`	The ID of the structure involved.
`issueId`	The ID of the JIRA issue involved.
`message`	More detailed message, may contain technical details.
`localizedMessage`	User-displayable message in the REST API user locale or JIRA default locale if the user is not authenticated.

Top(see page 1087)

Structure Resources

GET /structure

```
GET $baseUrl/rest/structure/2.0/structure
GET $baseUrl/rest/structure/2.0/structure?
name=$name&permission=$permission&withPermissions=$withPermissions&withOwner=$withOwn
er&limit=100
```

838 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/StructureError.html

A list of all structures visible to the REST API user. Optionally, the result can be filtered by name or user's access level. By default, permission rules and owners are not included, you should use query parameters if you want them to be included.

> **Who can access this resource**
>
> All users who have access to the Structure Plugin(see page 955). The returned list contains only structures to which the REST API user has at least View(see page 543) access level.

Request
Query parameters:

`name`	If present, the returned list will contain only structures which names contain the specified string (case insensitive).
`permission`	If present, the returned list will contain only structures to which the REST API user has the specified access level(see page 0). Must be equal to one of the names of the Permission Level[839] enum constants, case-insensitive. NONE[840] is treated in the same way as VIEW[841]. Please note that Control permission is represented by the ADMIN[842] enumeration constant.
`withPermissions`	If `true`, permission rules will be included in the response. Default is `false`.
`withOwner`	If `true`, owner will be included in the response. Default is `false`.
`archived`	If `true`, the returned list can also contain archived structures. Default is `false`.
`limit`	If specified, must be a number. Defines the maximum number of structures to return.

Each of the filter parameters `name`, `permission`, or `issueId` can be specified only once, otherwise the first is used. Different parameters are combined with AND.

HTTP headers:

`Content-Type`	Should be one of `application/json`, `application/xml`.

839 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/permissions/PermissionLevel.html
840 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html#NONE
841 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html#VIEW
842 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html#ADMIN

`Accept`	Should be one of `application/json`, `application/xml`.

Response
Success

`200 OK`	Response entity contains the only field, `structures`, which contains the list of the structure objects, sorted by name.	`application/json`, `application/xml`

Example 1: all structures

```
GET $baseUrl/rest/structure/2.0/structure
```


```
{
  "structures": [
    {
      "id": 1,
      "name": "Global Structure",
      "description": "Initial general-purpose structure.",
      "editRequiresParentIssuePermission": true
    },
    {
      "id": 102,
      "name": "Test plan",
      "description": "Test plan #3",
      "readOnly": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2"
    }
  ]
}
```


Example 2: only "Test plan"

```
GET $baseUrl/rest/structure/2.0/structure?name=test+plan
```

```
{
  "structures": [
    {
      "id": 102,
      "name": "Test plan",
      "description": "Test plan #3",
      "readOnly": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2"
    }
  ]
}
```

Example 3: structures that the user can edit with permissions and owners shown

```
GET $baseUrl/rest/structure/1.0/structure?permission=edit&withPermissions=true&withOwner=true
```

```
{
  "structures": [
    {
      "id": 1,
      "name": "Global Structure",
      "description": "Initial general-purpose structure.",
      "editRequiresParentIssuePermission": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1",
      "permissions": [
        {
          "rule": "set",
          "subject": "group",
          "groupId": "jira-users",
          "level": "edit"
        },
        {
          "rule": "set",
          "subject": "projectRole",
          "projectId": 10010,
          "roleId": 10010,
          "level": "none"
        },
        {
          "rule": "apply",
          "structureId": 101
        }
      ],
      "owner": "user:jsmith"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2",
      "owner": "user:admin"
    }
  ]
}
```


Example 4: require XML representation
Note that the same can be achieved by specifying `application/xml` in the `Accept` HTTP header.

```
GET $baseUrl/rest/structure/1.0/structure.xml?name=test+plan
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<structureList>
  <structures>
    <structure>
      <id>100</id>
      <name>Test plan</name>
      <description>Test plan #1</description>
    </structure>
  </structures>
</structureList>
```

Error

`400 Bad Request`	`permission` parameter is set to an unknown value, or request is invalid for other reasons. In the first case, response contains error entity, in the second it is empty.	`application/json`, `application/xml`
`403 Forbidden`	If Structure Plugin is not accessible to the REST API user, or if issue with ID `issueId` does not exist or the REST API user does not have enough permissions to access it. Response contains error entity.	`application/json`, `application/xml`
`404 Not Found`	If `issueId` is not an integer. Response entity contains a standard JIRA error HTML page.	`text/html`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 962) operation may be in progress.	

Other return codes[843] are possible under the normal rules of HTTP communication.

Top(see page 1087)

POST /structure

```
POST $baseUrl/rest/structure/2.0/structure
```

Create(see page 534) an empty structure by POSTing to this resource.

843 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

Who can access this resource

Only logged in users who have access to the Structure Plugin(see page 955) and a permission to create structures(see page 955).

Request
Request entity should contain the new structure(see page 1088). Structure name, `name`, must be present and non-empty. Fields `id`, `readOnly`, and `owner` are ignored. All rules in `permissions` are validated according to their respective rule types(see page 1089).

Please note that this resource accepts only JSON structure representation.

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response
Success

`201 Created`	Response entity contains the created structure with fields, including `permissions` and `owner`.	`application/json`, `application/xml`

Example 1: minimal structure

```
POST $baseUrl/rest/structure/2.0/structure
```

Request entity

```
{
  "name":"Test plan"
}
```

Response entity

```
{
  "id": 104,
  "name": "Test plan",
  "description": "",
  "permissions": [],
  "owner": "user:admin"
}
```

Example 2: structure with some permissions

```
POST $baseUrl/rest/structure/2.0/structure
```

Request entity	Response entity

{
"name":"Structure with some permissions",
"editRequiresParentIssuePermission": "true",
"permissions":[
{
"rule":"apply",
"structureId":102
}
]
}
	{
	"id": 105,
	"name": "Structure with some permissions",
	"description": "",
	"editRequiresParentIssuePermission": true,
	"permissions": [
	{
	"rule": "apply",
	"structureId": 102
	}
	],
	"owner": "user:admin"
	}

Error

`400 Bad Request`	Structure data is not well-formed (syntax error) or invalid (semantic error.) Not well-formed structure data examples: request JSON is syntactically incorrect; JSON contains unknown field; `name` is not present or empty; `permissions` list contains a *set* rule with `level` set to an invalid value. Invalid structure example: `permissions` list contains a rule that fails validation. Response entity contains error. Problems with *apply* rule usually have `structureId` to indicate the invalid reference.	`application/json`, `application/xml`
`403 Forbidden`	If REST API user is not logged in or does not have permissions to access Structure Plugin or to create structures. Response contains error entity.	`application/json`, `application/xml`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 962) operation may be in progress.	

Other return codes[844] are possible under the normal rules of HTTP communication.

Top(see page 1087)

GET /structure/{id}

```
GET $baseUrl/rest/structure/2.0/structure/$id
GET $baseUrl/rest/structure/2.0/structure/$id?
withPermissions=$withPermissions&withOwner=$withOwner
```

This resource allows to obtain structure details(see page 1088) for the particular structure. By default, `permissions` and `owner` are not included, use query parameters to include them.

> **Who can access this resource**
>
> All users who have access to the Structure Plugin(see page 955). To access the particular structure, the user has to have at least View(see page 543) access level.

Request
Path parameter:

`id`	the ID of the structure

Query parameters:

`withPermissions`	If `true`, permission rules will be included in the response. Default is `false`.
`withOwner`	If `true`, owner will be included in the response. Default is `false`.

HTTP headers:

`Content-Type`	Should be one of `application/json`, `application/xml`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response
Success

844 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

200 OK	Response entity contains the created structure along with all of its fields. Field `permissions` is included if the REST API user has Control(see page 543) permission on this structure. Field `owner` is included if the REST API user is either the owner of this structure or has Browse Users permission[845].	`application/json`, `application/xml`

Example 1: retrieve structure with ID 100 without permissions and owner

```
GET $baseUrl/rest/structure/2.0/structure/100
```

```
{
  "id": 100,
  "name": "Test plan",
  "description": "Test plan #1"
}
```

Example 2: permissions and owner are requested to be included, but only owner is shown, because the user has only View access as indicated by `readOnly`

```
GET $baseUrl/rest/structure/2.0/structure/102?withOwner=true&withPermissions=true
```

```
{
  "id":102,
  "name":"Test plan",
  "description":"Test plan #3",
  "readOnly":true,
  "owner":"user:admin"
}
```

Example 3: XML representation may be requested in the request URL instead of the `Content-Type` HTTP header

```
GET $baseUrl/rest/structure/2.0/structure/102.xml
```

```
<structure>
  <id>102</id>
  <name>Test plan</name>
  <description>Test plan #3</description>
  <readOnly>true</readOnly>
</structure>
```

Error

845 https://confluence.atlassian.com/display/JIRA/Managing+Global+Permissions

`400 Bad Request`	One of the query parameters is too long.	
`403 Forbidden`	If REST API user does not have permissions to access Structure Plugin or does not have at least View(see page 543) permission on this structure. Response contains error entity.	`application/json`, `application/xml`
`404 Not Found`	If `id` is not an integer in `1..2^63-1`. Response entity contains a standard JIRA error HTML page.	`text/html`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 962) operation may be in progress.	

Other return codes[846] are possible under the normal rules of HTTP communication.

Top(see page 1087)

POST /structure/{id}/update

```
POST $baseUrl/rest/structure/1.0/structure/$id/update
```

Update one or several fields of a structure by POSTing to this resource.

> **Who can access this resource**
>
> Only logged in users who have access to the Structure Plugin(see page 955) and Control(see page 543) permission on this structure.

Request
Request entity should contain those structure fields(see page 1088) that need to be changed. Non-present fields will not be changed (for this user; `readOnly` may change for other users as a result of changing `permissions`.) Fields `id`, `readOnly`, and `owner` are ignored.

Please note that `permissions` field is modified as a whole, so to add a rule, you have to provide the new list of rules in the proper order.

If `permissions` field is present, all rules are validated according to their respective rule types(see page 1089).

846 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

Please note that this resource accepts only JSON structure representation.

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response
Success

`200 OK`	Response entity contains the updated structure with all fields, including `permissions` and `owner`.	`application/json`, `application/xml`

Example 1: change description of the Global Structure

```
POST $baseUrl/rest/structure/1.0/structure/1/update
```

Request entity

```
{
  "description":"Company-wide
structure providing the Big
Picture."
}
```

Response entity

```
{
  "id":1,
  "name":"Global Structure",
  "description":"Company-wide structure
providing the Big Picture.",
  "editRequiresParentIssuePermission":true,
  "permissions":[
    {
      "rule":"set",
      "subject":"anyone",
      "level":"view"
    },
    {
      "rule":"set",
      "subject":"group",
      "groupId":"jira-users",
      "level":"edit"
    },
    {
      "rule":"set",
      "subject":"group",
      "groupId":"jira-administrators",
      "level":"admin"
    }
  ]
}
```

Example 2: changing permission rules

```
POST $baseUrl/rest/structure/1.0/structure
```

Request entity

```
{
  "permissions":[
    {
      "rule":"set",
      "subject":"group",
      "groupId":"jira-users",
      "level":"edit"
    },
    {
      "rule":"apply",
      "structureId":101
    }
  ]
}
```

Response entity

```
{
  "id": 105,
  "name": "Structure with some
permissions",
  "description": "",
  "editRequiresParentIssuePermission":
true,
  "permissions": [
    {
      "rule": "set",
      "subject": "group",
      "groupId": "jira-users",
      "level": "edit"
    },
    {
      "rule": "apply",
      "structureId": 101
    }
  ],
  "owner": "user:admin"
}
```

Error

`400 Bad Request`	Structure data is not well-formed (syntax error) or invalid (semantic error.) Not well-formed structure data examples: request JSON is syntactically incorrect; JSON contains unknown field; `permissions` list contains a *set* rule with `level` set to an invalid value. Invalid structure example: `permissions` list contains a rule that fails validation. Response entity contains error. Responses to problems with an *apply* rule usually have `structureId` to indicate the invalid reference.	`application/json`, `application/xml`.
`403 Forbidden`	If REST API user is not logged in, does not have permissions to access Structure Plugin, or does not have Control(see page 543) access level to this structure. Response contains error entity.	`application/json`, `application/xml`

`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 962) operation may be in progress.	

Other return codes[847] are possible under the normal rules of HTTP communication.

Top(see page 1087)

DELETE /structure/{id}

Deletes(see page 555) the designated structure.

> **Who can access this resource**
>
> Only logged in users who have access to the Structure Plugin(see page 955) and Control(see page 543) permission on this structure.

Request
Path parameter:

`id`	the ID of the structure

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be absent or equal to one of `application/json`, `application/xml`.

Response
Success

`200 OK`	Contains an object with the only field `empty` with value `true`.	`application/json`, `application/xml`

Note: it should have been `204 No content` instead, but there were reports of some browsers (Firefox) incorrectly processing such results, so it's as it is.

Example

847 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

```
DELETE $baseUrl/rest/structure/1.0/structure/108
```

```
{
  "empty": true
}
```

Error

`403 Forbidden`	If REST API user is not logged in, does not have permissions to access Structure Plugin, or does not have Control(see page 543) access level to this structure. Response contains error entity.	`application/json`, `application/xml`
`404 Not Found`	If `id` is not an integer in `1..2^63-1`. Response entity contains a standard JIRA error HTML page.	`text/html`
`404 Not Found`	If `id` is an integer in `1..2^63-1`, but the structure with the specified `id` does not exist or the user does not have View(see page 543) access level to it.	`application/json`, `application/xml`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 962) operation may be in progress.	

Other return codes[848] are possible under the normal rules of HTTP communication.

Top(see page 1087)

10.6.4.4 Forest Resource

Forest Resource is responsible for serving forests and forest updates and receiving the forest actions (change commands) from the client.

848 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

Retrieving Forest

Request

```
GET $baseUrl/rest/structure/2.0/forest/latest?s=$forestSpec
POST $baseUrl/rest/structure/2.0/forest/latest
```

Returns the hierarchical issue list (forest) of the specified structure.

Parameters:

$forestSpec	*required*	The URL-encoded JSON representation of ForestSpec[849]. See also: RestForestSpec[850].
POST content	*required*	While GET method is preferred, POST is more robust because there's no risk of exceeding URL length with large forest specifications. The content is the same JSON object (but not URL-encoded, obviously).

Example:

```
GET /rest/structure/2.0/forest/latest?s={%22structureId%22:113}
```

Retrieves latest forest for structure #113.

Response

```
{
  "spec":{"structureId":113},
  "formula":"10394:0:4/356,10332:0:14707,10374:1:5/240,10348:2:14717",
  "itemTypes":{
    "4":"com.almworks.jira.structure:type-generator",
    "5":"com.almworks.jira.structure:type-folder"
  },
  "version":{
    "signature":-1659607419,
    "version":1
  }
}
```

In this reply, the most important part is "formula", which contains serialized information about the forest.

849 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestSpec.html
850 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/rest/RestForestSpec.html

Each component (delimited by comma) represents a row and looks like this: 10374:1:5/240:3. In this example, the numbers are:

- 10374 is the row ID
- 1 is the row depth
- 5/240 is the item identity
- 3 (optional) is a row semantic

If the row contains an issue, it's just issue ID; otherwise, it has the format of <item type>/<long item id>, or <item type>//<string item id>. Item type is a number, which is expanded in the "itemTypes" map in the reply. Row semantic is optional extra data about the row that is used by Structure internally.

Changing Forest

To change a forest, you POST one or more change actions to `/forest/update` resource. Each action is a serialized version of `ForestAction` – for more information about the actions, see Changing Structure Content(see page 1041).

```
POST $baseUrl/rest/structure/2.0/forest/update
```

Parameters:

```
{
  "spec": { "structureId": <id> }, // use structure ID
  "version": { "signature": <signature>, "version": <version> },  // use last seen signature and version
  "actions": [
     {
         "action": "add",
         "under": 0,        // at the top level
         "after": 123,      // after row ID 123 (not issue iD!)
         "before": 456,     // before row ID 456
         "forest": "-100:0:10001"   // insert issue 10001, -100 is the temporary row ID which will be mapped into the real row ID when the method returns
     },
     {
         "action": "move", // works like previously, only row IDs instead of issue IDs
         "rowId": 123,
         "under": 456,
         "after": 0,
         "before": 124
     },
    {
       "action": "remove",
       "rowId": 442
     }
  ]
}
```

10.6.4.5 Item Resource

Item resource is used to create new items and update existing items.

Creating a New Item

The following request is used to create a new item (issue, folder or other type) and insert it into a forest.

```
POST $baseUrl/rest/structure/2.0/item/create
```

This request should upload a specification of the creation action and coordinates of where to put the result.

Example

```
{
  "item": {
    "type": "com.almworks.jira.structure:type-folder",
    "values": { "summary": "New folder name" }
  },
  "forest": {
    "spec": { "structureId": 128 },
    "version": {
      "signature": 0,
      "version": 0
    }
  },
  "items": {
    "version": {
      "signature": 0,
      "version": 0
    }
  },
  "rowId": -100,
  "under": 0,
  "after": 0,
  "before": 0,
  "parameters": {}
}
```

Parameters

Parameter (see example above)	Meaning
`item`	Defines the item being created.

Parameter (see example above)	Meaning
`item.type`	Item type (complete key of the module that provides this item's main functionality.) Use `com.almworks.jira.structure:type-folder` for folders and `com.almworks.jira.structure:type-issue` for issues. See also: CoreItemTypes[851]
`item.values`	A set of values for the new item. The specific fields depend on the item. For a folder, it is "summary". For other items, see examples below.
`forest.spec`	Forest specification of the forest that will receive the new item. See ForestSpec[852] and RestForestSpec[853].
`forest.version`	Last known version of the forest. The reply to this call will contain the update to that version. Use zero version (as in example) to receive full forest.
`items.version`	Last known version of instance items set. The reply to this call will contain an update to the known items. Use zero version (as in example) to receive full update.
`rowId`	Temporary ID assigned to the created issue. Must be negative. You can use `-100` in most cases.
`under` / `after` / `before`	Forest coordinates to insert the new item into. See Forest Resource(see page 1105).

Specific parameters for main item types

Folder
This is the example of `item` parameter for a new folder:

```
"item": {
  "type": "com.almworks.jira.structure:type-folder",
  "values": { "summary": "New folder name" }
}
```

The only parameter sent is the folder name.

Issue
This is the example of `item` parameter for a new issue:

851 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/CoreItemTypes.html
852 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestSpec.html
853 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/rest/RestForestSpec.html

```
"item": {
  "type": "com.almworks.jira.structure:type-issue",
  "values": {
    "issue": {
      "summary": "issue summary"
    },
    "pid": 10000,
    "issuetype": "3",
    "mode": "new",
  }
}
```

The above are the minimal fields needed to create a new issue. Note that `pid` is a number, but `issuetype` is a string.

Reply Example

The following is an example of a reply.

```
{
  "successfulActions": 1,
  "itemId": "com.almworks.jira.structure:type-issue/10100",
  "oldRowIds": [-100],
  "newRowIds": [61],

  "forestUpdates": [...],
  "itemsUpdate": {...}
}
```

Most important fields are `itemId` and `newRowIds`. More on the return fields:

Field	Explanation
`successfulActions`	A number of actions successfully performed by the server. In this case, it's either 0 or 1.
`itemId`	The ID of the newly created item. See ItemIdentity[854].
`oldRowIds` / `newRowIds`	Provides mapping from the temporary row IDs used for uploading the action and the real row IDs obtained after the item was inserted.
`forestUpdates`	Changes to the forest since the version passed in the request.
`itemsUpdate`	Changes to the items set since the version passed in the request.

Updating an Existing Item

The following request is used to update an existing item (issue, folder or other type).

854 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/ItemIdentity.html

```
POST $baseUrl/rest/structure/2.0/item/update
```

Example of the request:

```
{
  "item": {
    "itemId": "10000",
    "values": {
      "summary": "New Summary"
    }
  },
  "items": {
    "version": { "signature": 0, "version": 0 }
  },
  "forest": {
    "spec": {
      "type": "clipboard"
    },
    "version": { "signature": 0, "version": 0 }
  }
}
```

Note that although the update does not depend on the forest, the low-level API in the current version requires the request to specify a forest spec and known version of items stream. If you don't need to maintain up-to-date items cache and not interested in updates to a forest where the item is located, just use empty version in **items** field and "clipboard" forest spec – like in this example.

Parameters

Parameter (see example above)	Meaning
`item.itemId`	The ID of the item. If it is just a number, like in the example, it is an issue ID. Note that it is still a **String** value that contains issue ID. Instead of a number, it can be a canonical notation of an ItemIdentity[855]. For example, to update a folder, use `"com.almworks.jira.structure:type-folder/123"` where 123 is the folder ID.

855 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/ItemIdentity.html

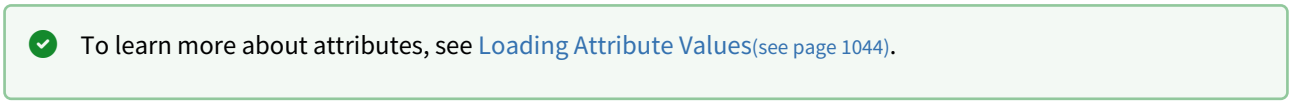
Parameter (see example above)	Meaning
`item.values`	A map of values to be updated. The keys are the same as when the item is created. For updating a folder, use `"summary"`.
`items.version`	Known version of the items stream. The response will contain an update based on that number. Use zeroes, as in example, when updated is not needed.
`forest.spec` and `forest.version`	Monitored forest spec and known version of that forest. The response will contain a forest update based on those values. When not needed, use a simple forest (like clipboard in this example) and zeroed version.

Reply

The reply is similar to the reply from calling `/create` method, defined above. A positive HTTP status tells that the item has been updated. There is no `"itemId"` in the response.

10.6.4.6 Value Resource

Value Resource is used to retrieve values of attributes for rows in a given forest.


To learn more about attributes, see Loading Attribute Values(see page 1044).

To retrieve values from Structure, you need a few things first:

- A forest specification (`forestSpec`) for the displayed forest – same as the one used in Forest Resource(see page 1105). Forest specification is needed even if the values do not depend on the forest.
- A list of row IDs for which the values should be loaded. Row IDs can be retrieved from Forest Resource before calling Value Resource.
- A list of attribute specifications. Some examples are given below.

Loading Values

To load values use the following call

```
POST $baseUrl/rest/structure/2.0/value
```

The request should come with JSON payload that specifies which values you are interested in.

Example

```
{
  "requests": [
    {
      "forestSpec": {
        "structureId": 123
      },
      "rows": [
        1820,
        1842,
        2122
      ],
      "attributes": [
        {
          "id": "summary",
          "format": "text"
        },
        {
          "id": "key",
          "format": "html"
        },
        {
          "id": "progress",
          "format": "number",
          "params": {
            "basedOn": "timetracking",
            "resolvedComplete": true,
            "weightBy": "equal"
          }
        }
      ]
    }
  ]
}
```

As you see in this example, a request body may contain one or more request, each for a specific matrix of several rows and several attributes. A value for each pair of a row and an attribute will be calculated.

Parameters

Parameter	Meaning
`requests[i].forestSpec`	Forest specification that produces the forest from which the rows are taken.
`requests[i].rows`	Array of row IDs for which values should be loaded.
`requests[i].attributes`	Array of attribute specifications that should be loaded for each row.

The example shows three attributes being loaded – plain text Summary, html-formatted Key and Progress based on time tracking. For more information about available system attributes, see javadocs for AttributeSpec[856] and CoreAttributeSpecs[857].

There is a simple way to learn the attribute spec that you need.

1. Configure a column that shows the needed value on the Structure Board.
2. Use your browser's Developer Tools and open Network tab.
3. Reload structure.
4. Look for a request to `/value` URL and see its input. Use JSON formatters for convenience.

Response

The response will contain one or more matrices with values for each pair of requested row and attribute. A list of rows is given separately. Then, for each requested attribute, a list of values is given.

856 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/AttributeSpec.html
857 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/CoreAttributeSpecs.html

```
{
  "responses": [
    {
      "forestSpec": {
        "structureId": 123
      },
      "rows": [
        1820,
        1842,
        2122
      ],
      "data": [
        {
          "attribute": {
            "id": "summary",
            "format": "text"
          },
          "values": [
            "Issue 1",
            "Folder 2",
            "Some Other Item 3"
          ],
          "trailMode": "INDEPENDENT",
          "trails": [ "", "", "" ]
        }
      ],
      "forestVersion": {
        "signature": -1385959428,
        "version": 1
      }
    }
  ],
  "itemTypes": {},
  "itemsVersion": {
    "signature": -558220658,
    "version": 1
  }
}
```


Parameters

Parameter	Meaning
`responses[i].forestSpec`	Requested forest spec, from which the rows are taken.
`responses[i].rows`	A list of row IDs for which the values are provided.
`responses[i].data[j].attribute`	The attribute specification for which the following values are calculated.

Parameter	Meaning
`responses[i].data[j].values`	Array of values. The value at k-th place corresponds to the row at k-th place in `responses[i].rows`.

⚠ If you are receiving value in any format other than `html`, you need to html-escape that value before adding it to the web page.

10.6.4.7 Attribute Subscription Resource

Attribute subscription resource is used to retrieve attribute values and updates to those values sequentially through a subscription.

ⓘ Attribute Subscription Resource is introduced in Structure 6.0. When working with an older version, use Value Resource(see page 1112).

To learn more about attributes, see Loading Attribute Values(see page 1044).

Subscriptions

A subscription represents an interest of a client code in a set of attributes for a set of rows in a particular forest. Just like when loading attributes through Value Resource(see page 1112), in order to create a subscription, you need:

- A forest specification (`forestSpec`) for the displayed forest.
- A list of row IDs for which the values should be loaded. Row IDs can be retrieved from Forest Resource before calling Value Resource.
- A list of attribute specifications. Some examples are given below and in the documentation for Value Resource(see page 1112).

The chosen rows and attributes of interest are called a "window". The subscription server keeps track of registered subscriptions and corresponding windows, and, upon requests, calculates the updated data for a window and sends the updates back to the client.

✓ Attribute subscription is the preferred way to continuously receive attribute values. For a one-off download, it's better to use Value Resource.

Base URL

Base URL for Attribute Subscription Resource is: `$baseUrl/rest/structure/2.0/attribute/subscription`

In the following documentation we will simply write `/attribute/subscription` meaning the full base URL above.

Common Parameters and Data Structures

There are a number of parameters that are used repeatedly in multiple methods in the Attribute Subscription Resource.

Input Query Parameters

These are passed as a part of URL. Neither of them are required.

Parameter Name	Type	Default	Meaning
`valuesUpdate`	Boolean	`false`	Indicates whether a client desires to receive the updated values for the current or updated window. This typically implies that a request to load these values will be executed.
`valuesTimeout`	Long	`1000ms`	Indicates the maximum time (in milliseconds) the server will wait for the values to be calculated before responding to the client. Applies only if `valuesUpdate` is true. The default can be adjusted through Advanced Configuration and Dark Features(see page 974).
`signature`	Integer	`0`	Together, signature and version define the last point of synchronization with the updates from the server. In the first request, the client should use zeros or not use them at all. Each successful response will include an updated signature and version, which client code should use the next time in order to get only the values that have changed.
`version`	Integer	`0`	

Data Structure: `SubscriptionWindow`

The `SubscriptionWindow` data structure represents a subscription window.

Field	Type	Meaning
`forestSpec`	Object: a forest specification, the same as in Forest Resource(see page 1105)	Defines the forest for which the values will be loaded.
`rows`	Array of Long numbers	Defines a list of rows to be loaded.
`attributes`	Array of Objects: AttributeSpec	Defines a list of attributes to be loaded.

Example:

```
{
  "forestSpec": { "structureId": 1 },
  "rows": [ 10, 15, 1, 27, 1001 ],
  "attributes": [
    { "id": "summary", "format": "text" },
    { "id": "key", "format": "html" },
    {
      "id": "progress",
      "format": "number",
      "params": {
        "basedOn": "timetracking",
        "resolvedComplete": true,
        "weightBy": "equal"
      }
    }
  ]
}
```

Data Structure: `SubscriptionData`

The `SubscriptionData` data structure is sent back from most of the REST calls to the Attribute Subscription Resource.

Field	Type	Meaning
`id`	Integer	The unique ID of the subscription.
`window`	`SubscriptionWindow`	The current (updated) window for the subscription.
`valuesUpdate`	`SubscriptionUpdate`	An update for the current subscription.

Data Structure: `SubscriptionUpdate`

A subscription update is returned as a part of `SubscriptionData` structure and carries the new values, versioning information, error information and loading progress information.

This structure is also used in the Poll Resource.

Field	Type	Meaning
`id`	Integer	The unique ID of the subscription.
`full`	Boolean	If true, this is a "full" or "total" update, meaning that all old values are obsolete. If false, it is an incremental update and includes only the updated values.

Field	Type	Meaning
`fromVersion`	`RestVersion`	Contains signature and version that were used to request the update.
`version`	`RestVersion`	Contains signature and version that should be used to request the next update.
`data`	Array of Objects	Each object represents values for a particular attribute.
`data[i].attribute`	AttributeSpec	Describes the attribute for which the values are provided.
`data[i].values`	Object	A map from row IDs (represented as strings) to the attribute values. A value may take multiple forms – it could be a number, a text, a boolean, an array or an object.
`data[i].outdated`	Array of Longs	Optional field that lists the values which are known to contain outdated values. (But the values are provided all the same, so that the client can show something while the updated values are being calculated.)
`stillLoading`	Boolean	If true, not all of the requested values have been loaded. The client can retry the request some time later.
`inaccessibleRows`	Array of Longs	Optional field that lists all requested rows that are not accessible by the user (for any reason). The client code should not continue requesting them.
`error`	`StructureError`	An object containing error code and other diagnostics in case of a problem.
`attributeErrors`	A list of Objects	An optional list of attribute-specific errors.

Resource Methods

Create Subscription

```
POST /attribute/subscription
```

Query parameters	`valuesUpdate`, `valuesTimeout`	Allow immediately loading some values for the created subscription.
Input data	`SubscriptionWindow`	Passed as the request body, the window defines the subscription parameters.

Response	`SubscriptionData`	Contains the ID of the newly created subscription, the used window, and some values if they were requested.

Retrieve Subscription or Values

```
GET /attribute/subscription/ID
```

Path parameters	ID	Identifies the subscription by ID.
Query parameters	`valuesUpdate`, `valuesTimeout`	Allow immediately loading some values for the created subscription.
	`signature`, `version`	Identify the previous version of the values that the client has seen to get incremental updates.
	`skipLoading`	A Boolean parameter: if `true`, then only the already calculated values will be loaded, without requesting the attribute subsystem to reload the values in the window.
Response	`SubscriptionData`	Contains all the subscription data and value updates if they were requested.

Update Subscription

Used to completely change the subscription's window.

```
PUT /attribute/subscription/ID
```

Path parameters	ID	Identifies the subscription by ID.
Query parameters	`valuesUpdate`, `valuesTimeout`	Allow immediately loading some values for the updated subscription.
	`signature`, `version`	Identify the previous version of the values that the client has seen to get incremental updates.
Input data	`SubscriptionWindow`	The updated subscription window.
Response	`SubscriptionData`	Contains all the subscription data and value updates if they were requested.

Patch Subscription

Used to partially change the subscription's window. Only the fields mentioned in the input `SubscriptionWindow` object will be updated.

```
POST /attribute/subscription/ID/patch
```

Path parameters	ID	Identifies the subscription by ID.
Query parameters	`valuesUpdate`, `valuesTimeout`	Allow immediately loading some values for the updated subscription.
	`signature`, `version`	Identify the previous version of the values that the client has seen to get incremental updates.
Input data	`SubscriptionWindow`	The object with the parts of the subscription window that need to be updated.
Response	`SubscriptionData`	Contains all the subscription data and value updates if they were requested.

Delete Subscription

```
DELETE /attribute/subscription/ID
```

Path parameters	ID	Identifies the subscription by ID.
Response	`HTTP 200`	Simple response with no data.

User Access

When a subscription is created, it is attached to the current user. Only this user will have access to the subscription.

Subscription Expiration and Caching

A subscription is considered a transient, cache-like object. It may be removed from the server at any time, even if the user has not requested it.

When a request is made about a subscription that no longer exists, the server responds with HTTP 404 error and an object describing the error in detail. The client code may decide to re-create a subscription at that point.

10.6.4.8 Configuration Resource

This page describes resources which allow you to manage the list of projects(see page 954) for which Structure is enabled and the global permissions(see page 954) of the Structure app.

> ⓘ Only logged in Jira administrators can access this resource.

Configuration resource belongs to **version 2.0** of the API, its path is `/rest/structure/2.0/configuration`. It contains the following resources:

/projects GET(see page 1122)	Get the list of projects for which Structure is enabled
/projects PUT(see page 1123)	Update the list of projects for which Structure is enabled
/projects/add POST(see page 1123)	Enable Structure for projects
/projects/remove POST(see page 1124)	Disable Structure for projects
/permissions GET(see page 1126)	Get Structure's global permissions
/permissions PUT(see page 1128)	Update Structure's global permissions
/permissions/{key} GET(see page 1129)	Get a particular global permission
/permissions/{key} PUT(see page 1130)	Update a particular global permission
/permissions/{key}/add POST(see page 1130)	Grant a global permission to user groups or project roles
/permissions/{key}/remove POST(see page 1131)	Revoke a global permission from user groups and project roles

Quick navigation:

- Permission keys(see page 1125)
- Permission configuration objects(see page 1125)
- Permission subjects(see page 1126)

Project Resources

GET /projects

Returns a JSON object which contains:

- a flag telling whether Structure is enabled for all projects, and
- the list of project IDs for which Structure is enabled.

Please note that even when Structure is enabled for all projects, the list of selected project IDs is still stored in the app configuration and returned by this resource.

```
GET $baseUrl/rest/structure/2.0/configuration/projects
```

A successful request returns a JSON object with two fields, for example:

```
{
  "enabledForAllProjects": false,
  "pickedProjectIds": [10000, 10100]
}
```

Top(see page 1121)

PUT /projects

Updates the "enabled for all projects" flag and/or the list of projects for which Structure is enabled.

```
PUT $baseUrl/rest/structure/2.0/configuration/projects
```

Accepts a JSON object with the following fields, at least one of which must be present:

- `enabledForAllProjects` – a boolean indicating whether Structure is enabled for all projects. If omitted, the setting is not changed.
- `pickedProjectIds` – an array of numbers with the IDs of the projects for which Structure is enabled. If omitted, the list of projects is not changed.

Please note that when Structure is enabled for all projects, the list of projects has no effect, but it is still stored in the app configuration and can be updated.

Example 1. Enable Structure for only two projects with IDs 10000 and 10100:

```
PUT $baseUrl/rest/structure/2.0/configuration/projects


{
  "enabledForAllProjects": false,
  "pickedProjectIds": [10000, 10100]
}
```

Example 2. Enable structure for all projects, don't change the list of projects:

```
PUT $baseUrl/rest/structure/2.0/configuration/projects


{ "enabledForAllProjects": true }
```

A successful request returns an "empty" JSON object:

```
{ "empty": true }
```

Top(see page 1121)

POST /projects/add

Adds one or more projects to the list of projects for which Structure is enabled.

```
POST $baseUrl/rest/structure/2.0/configuration/projects/add
```

Accepts a JSON array of numbers, all of which must be existing project IDs. At least one project ID must be supplied.

Please note that when Structure is enabled for all projects, the list of projects has no effect, but it is still stored in the app configuration and can be updated.

Example. Add project with IDs 10200 and 10300 to the list of projects:

```
POST $baseUrl/rest/structure/2.0/configuration/projects/add


[10200,10300]
```

A successful request returns a JSON object with a single boolean flag indicating whether the list of projects was updated (i.e. it did not contain at least one of the passed project IDs before the call):

```
{ "updated": true }
```

Top(see page 1121)

POST /projects/remove

Removes one or more projects from the list of projects for which Structure is enabled.

```
POST $baseUrl/rest/structure/2.0/configuration/projects/remove
```

Accepts a JSON array of numbers. Non-existent project IDs are allowed. At least one project ID must be supplied.

Please note that when Structure is enabled for all projects, the list of projects has no effect, but it is still stored in the app configuration and can be updated.

Example. Remove project with ID 10200 from the list of projects:

```
POST $baseUrl/rest/structure/2.0/configuration/projects/remove


[10200]
```

A successful request returns a JSON object with a single boolean flag indicating whether the list of projects was updated (i.e. it contained at least one of the passed project IDs before the call):

```
{ "updated": true }
```

Top(see page 1121)

Permission Resources

Permission Keys

The following global permissions are available:

Key	Permission
`use`	Use Structure. See Who Has Access to the Structure(see page 955).
`createStructure`	Create new structures. See Changing Permission to Create New Structures(see page 955).
`synchronization`	Configure synchronizers.
`automation`	Configure automation (generators and effectors). See Changing Permission to Access Automation(see page 956).
`configureGenerators`	Configure generators. The "automation" permission is also required.
`configureEffectors`	Configure effectors. The "automation" permission is also required. See Changing Permissions to Configure and Run Effectors(see page 956).
`executeEffectors`	Execute effectors.
`executeEffectorsOnQueries`	Execute effectors on query results.

Top(see page 1121)

Permission Configuration Objects

The configuration of each global permission is represented by a JSON object with two fields:

Field	Type	Description
`allowedForAnyone`	boolean	Indicates that a permission is granted to anyone.
`subjects`	array of objects	The list of permission subjects to which the permission is granted.

The flag and the list are stored independently. If a permission is granted to anyone, the list of permission subjects has no effect, but it is still stored in the Structure app configuration and can be updated.

Also, even if a global permission is granted to anyone, there can be other permission-specific restrictions. For example, to configure generators in a structure, a user needs the "Automate" access level to that structure.

Top(see page 1121)

Permission Subjects

Global permissions can be granted to user groups and project roles. Each permission subject is represented by a JSON object with the following fields:

Field	Type	Description
`subject`	string	The type of the subject. Must be either `"group"` or `"projectRole"`. Required.
`groupId`	string	Group name. Required for user groups.
`roleId`	number	Project role ID. Required for project roles.
`projectId`	number	Project ID. Required for project roles. 0 means "any project".

Example 1. The `jira-administrators` group:

```
{
  "subject": "group",
  "groupId": "jira-administrators"
}
```

Example 2. Role with ID 10002 in any project:

```
{
  "subject": "projectRole",
  "projectId": 0,
  "roleId": 10002
}
```

Top(see page 1121)

GET /permissions

Returns all global permissions as a JSON object.

```
GET $baseUrl/rest/structure/2.0/configuration/permissions
```

In the returned JSON object, keys are permission keys(see page 1125) and values are permission configuration objects(see page 1125). Example:

```
{
  "use": {
    "allowedForAnyone": true,
    "subjects": []
  },
  "createStructure": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-software-users"
      }
    ]
  },
  "synchronization": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-administrators"
      }
    ]
  },
  "automation": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-administrators"
      },
      {
        "subject": "projectRole",
        "projectId": 0,
        "roleId": 10002
      }
    ]
  },
  "configureGenerators": {
    "allowedForAnyone": true,
    "subjects": []
  },
  "configureEffectors": {
    "allowedForAnyone": true,
    "subjects": []
  },
  "executeEffectors": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "projectRole",
        "projectId": 10000,
```

```
        "roleId": 10100
      },
      {
        "subject": "projectRole",
        "projectId": 10100,
        "roleId": 10100
      }
    ]
  },
  "executeEffectorsOnQueries": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-administrators"
      },
      {
        "subject": "projectRole",
        "projectId": 0,
        "roleId": 10002
      }
    ]
  }
}
```

Top(see page 1121)

PUT /permissions

Updates one or more global permissions.

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions
```

Accepts a JSON object where keys are permission keys(see page 1125) and values are permission configuration objects(see page 1125). Omitted global permissions are not changed. For each permission, either field can be omitted, but at least one field must be supplied.

Example. Allow anyone to use Structure, but allow only the "jira-software-users" group to create new structures:

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions


{
  "use": {
    "allowedForAnyone": true
  },
  "createStructure": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-software-users"
      }
    ]
  }
}
```

A successful request returns an "empty" JSON object:

```
{ "empty": true }
```

Top(see page 1121)

GET /permissions/{key}

Returns a permission configuration object(see page 1125) for the given permission key(see page 1125).

```
GET $baseUrl/rest/structure/2.0/configuration/permissions/$key
```

Example. Get the configuration object for the permission to create new structures:

```
GET $baseUrl/rest/structure/2.0/configuration/permissions/createStructure

{
  "allowedForAnyone": false,
  "subjects": [
   {
      "subject": "group",
      "groupId": "jira-software-users"
    }
  ]
}
```

Top(see page 1121)

PUT /permissions/{key}

Updates the configuration of the given permission key(see page 1125).

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions/$key
```

Accepts a permission configuration object(see page 1125). Either field can be omitted, but at least one field must be supplied.

Example 1. Allow anyone to use Structure:

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions/use

{
  "allowedForAnyone": true
}
```

Example 2. Allow only Jira administrators and project administrators (role ID 10002) in any project to configure automation:

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions/automation

{
  "allowedForAnyone": false,
  "subjects": [
    {
      "subject": "group",
      "groupId": "jira-administrators"
    },
    {
      "subject": "projectRole",
      "projectId": 0,
      "roleId": 10002
    }
  ]
}
```

A successful request returns an "empty" JSON object:

```
{ "empty": true }
```

Top(see page 1121)

POST /permissions/{key}/add

Adds user groups and/or project roles to the list of permission subjects for the given permission key(see page 1125).

```
POST $baseUrl/rest/structure/2.0/configuration/permissions/$key/add
```

Accepts a JSON array of one or more permission subjects(see page 1126). If the permission is granted to anyone, the list of permission subjects has no effect, but it is still stored in the Structure app configuration and can be updated.

Example. Grant project administrators (role ID 10002) in any project the permission to execute effectors on query results:

```
POST $baseUrl/rest/structure/2.0/configuration/permissions/executeEffectorsOnQueries/
add

[
  {
    "subject": "projectRole",
    "projectId": 0,
    "roleId": 10002
  }
]
```

A successful request returns a JSON object with a single boolean flag indicating whether the list of permission subjects was updated (i.e. it did not contain at least one of the passed subjects before the call):

```
{ "updated": true }
```

Top(see page 1121)

POST /permissions/{key}/remove

Removes user groups and/or project roles from the list of permission subjects for the given permission key(see page 1125).

```
POST $baseUrl/rest/structure/2.0/configuration/permissions/$key/remove
```

Accepts a JSON array of one or more permission subjects(see page 1126). If the permission is granted to anyone, the list of permission subjects has no effect, but it is still stored in the Structure app configuration and can be updated.

Example. Revoke the global permission to configure synchronizers from Jira Software users:

```
POST $baseUrl/rest/structure/2.0/configuration/permissions/synchronization/remove

[
  {
    "subject": "group",
    "groupId": "jira-software-users"
  }
]
```

A successful request returns a JSON object with a single boolean flag indicating whether the list of permission subjects was updated (i.e. it contained at least one of the passed subjects before the call):

```
{ "updated": true }
```

Top(see page 1121)

10.6.5 Structure JavaScript API Reference

Structure's JavaScript API provides ways to extend the client-side functionality of the Structure plugin.

- JavaScript API Functions(see page 1132) — This page lists static functions exposed by the Structure API.
- JavaScript API Classes(see page 1134) — Structure Javascript API provides a number of classes to be used as base for your own column type implementations. This should be done using subClass() method.
    - Column Class(see page 1135) — A subclass of Column class represents column objects of a specific type.
    - ColumnConfigurator Class(see page 1139) — ColumnConfigurator class encapculates everything related to column type configuration.
    - ColumnOption Class(see page 1141) — ColumnOption class represents a single column configuration parameter.
    - ColumnType Class(see page 1144) — ColumnType class represents column type.
    - ItemDetailsProvider Class(see page 1145) — ItemDetailsProvider class is the extension point for item details.
- JavaScript API Objects(see page 1149) — This page lists objects exposed by the Structure API.
    - ItemDetailsBridge Object(see page 1149) — ItemDetailsBridge provides api for interaction with the Structure app and item details lifecycle.

10.6.5.1 JavaScript API Functions

This page lists static functions exposed by the Structure API.

window.almworks.structure.api.subClass(className, superclass, prototype)

Creates a subclass of a specific class. Returns a constructor function that will create the instances of the class.

This function provides light-weight polymorphism for the purposes of extending Structure's classes(see page 1134).

Parameters

`className`	*string*	Class name as string (optional, used for friendly instance names in debugger)
`superclass`	*Object*	Superclass reference
`prototype`	*Object*	Subclass prototype

The returned value – class constructor – takes a single optional `options` parameter.

The prototype may contain a special `init()` initializer method, which is called when an instance is being constructed. Superclass' `init()` method is called before subclass' method. Options that were passed to the constructor are passed through to the initializer.

Example

```
var MyClass = window.almworks.structure.api.subClass('MyClass', BaseClass, {
  init: function(options) {
   ...
  },

  someMethod: function() {
   ...
  }
});

var options = { ... };
var instance = new MyClass(options);
```

window.almworks.structure.api.registerColumnType(type, key)

Registers a new column type. If you're extending Structure by adding a new type of column to the grid, the type must be registered from your additional JavaScript web resource.

Column types are identified by a unique key, which is recorded in the view(see page 1132) specification, along with the type-specific parameters and column name.

Parameters

`type`	*Object*	A `ColumnType` instance, implementing a specific column type – see ColumnType Class(see page 1144)
`key`	*string*	Column type key (can also be array of strings if the type can handle multiple variations of a column specification)

Example

```
window.almworks.structure.api.registerColumnType(new MyColumnType(),
'com.acme.structure.awesome-column');
```

We recommend using a unique key that has low chance of conflicting with column types provided by other, independent developers. A good approach is to have Java-like package notation for the keys.

window.almworks.structure.api.registerColumnGroup(options)

Registers a new column type group. Column groups are used in the "Add Column" panel to group column configuration presets, provided by column types.

Parameters

`options.groupKey`	*string*	Group key. The same key should be returned by all `ColumnConfigurator`'s `getGroupKey()` method for all column types which need to appear in this particular group.
`options.title`	*string*	Group title
`options.order`	*number*	Group order is used to sort groups. Order value for groups provided by Structure are 100, 200, 300 and so on.

Example

```
window.almworks.structure.api.registerColumnGroup({
  groupKey: 'com.acme.structure.colgroup', title: 'Acme Columns', order: 50
});
```

window.almworks.structure.api.registerItemDetailsProvider(itemType, ProviderClass)

Registers item details support for the given item type.

Parameters

`itemType`	*string*	Item type for which details are registered
`ProviderClass`	*Object*	Subclass of ItemDetailsProvider[858] that defines details behavior

Example

```
var api = window.almworks.structure.api;
var MilestoneDetails = api.subClass('MilestoneDetails', api.ItemDetailsProvider, {
  ...
});

api.registerItemDetailsProvider('com.acme.my-plugin:type-milestone',
MilestoneDetails);
```

10.6.5.2 JavaScript API Classes

Structure Javascript API provides a number of classes to be used as base for your own column type implementations. This should be done using subClass()(see page 0) method.

- Column Class(see page 1135) — A subclass of Column class represents column objects of a specific type.

858 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class

- ColumnConfigurator Class(see page 1139) — ColumnConfigurator class encapculates everything related to column type configuration.
- ColumnOption Class(see page 1141) — ColumnOption class represents a single column configuration parameter.
- ColumnType Class(see page 1144) — ColumnType class represents column type.
- ItemDetailsProvider Class(see page 1145) — ItemDetailsProvider class is the extension point for item details.

Column Class

window.almworks.structure.api.Column

A subclass of Column class represents column objects of a specific type.Columns need to be subclassed for a particular column type implementation. You can override methods while subclassing to modify the default behavior.

Example

```
var api = window.almworks.structure.api;
var MyColumn = api.subClass('MyColumn', api.Column, {
  init: function() {
    ...
  },
  getCellViewHtml: function() {
    return '<div> ... </div>';
  }
});
```

Properties

`context`
Contains context information about where the column is used. See The Column Context(see page 1055) for more information.

`spec`
Contains column specification object. Specification object is serialized as a part of the overall view specification and stored on the server and in the browser's local storage. See Column Specifications(see page 1054) for more information.

Methods

`init(options)`
Initializer method.

`getCellValueHtml(renderingParameters)`
Returns HTML that is displayed in the grid cell for a specific issue. The HTML should contain the value provided by this column. Structure will also wrap the value in decorative elements – this could be overridden by providing `getCellViewHtml()` method.

Parameters

renderingParameters.getAttributeValue()	returns current row's attribute value

renderingParameters.getRowId()	returns current row's id
renderingParameters.getItemId()	return current row's item id

Example

```
var Template = require('almworks/util/Template');
var cellTemplate = new Template('<span class="acme-field">{awesomefield}</span>');
getCellValueHtml: function(rp) {
  return cellTemplate.renderHtml({ awesomefield: rp.getAttributeFieldValue({id:
'com.acme.awesome-data', format: 'text'}) });
}
```

`getCellViewHtml(renderingParameters)`
Returns customized HTML that is displayed in the grid cell for a specific issue. By default, calls `getCellValueHtml()` and wraps the retrieved value into the default Structure style. Can be overridden to allow higher degree of control over the cell appearance.

Parameters

renderingParameters.getAttributeValue()	returns current row's attribute value
renderingParameters.getRowId()	returns current row's id
renderingParameters.getItemId()	return current row's item id

`collectRequiredAttributes(attributeSet)`
Lets column request attributes that are needed for rendering. The attributes are provided on the server side by AttributeLoaderProvider[859].

Parameters

attributeSet.requireAttribute(attributeSpec,forestSpec)	Method for collecting required attributes. Parameters are: • `attributeSpec` is the attribute specification object • `forestSpec` is the forest specification for the forest, from which attribute should be loaded **(optional)**

About Attribute Specs

AttributeSpec defines the attribute and format to be loaded. See Loading Attribute Values(see page 1044) for more information on attributes.

Some of the attributes are shown below. You can also define your own attribute, calculate it on the server side and request from your column.

859 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/loader/AttributeLoaderProvider.html

About Forest Spec

Forest specification is optional. When used, it allows you to get attribute value from a different forest – however, it must be related to the forest being displayed, otherwise it will not have the same rows.

For example, you can specify a forest specification with some transformation to display values from there in the untransformed forest. There are also two special values for `forestSpec`:

- `'displayed'` is the default value, meaning "use the forest that is being displayed"
- `'unfiltered'` means "use the same forest, but remove all filters that are coming at the end of transformation chain"

Example

```
collectRequiredAttributes: function(attributeSet) {
  attributeSet.requireAttribute({id: 'key', format: 'text'});
  attributeSet.requireAttribute({
    id: 'sum',
    format: 'number',
    params: {
      id: 'customfield',
      format: 'number',
      params: {
        fieldId: 10010
      }
    }
  }, 'unfiltered');
  attributeSet.requireAttribute({id: 'com.mycompany.work-stats', format: 'json'});
}
```

Some of the attributes provided by Structure:

Attribute Spec	Example	Description
`{id: <jira-field-id>, format: 'html'}`		The HTML representation of a JIRA issue field value, as seen on the issue page or in the Issue Navigator. Structure allows non-issue items also have these values. `<jira-field-id>` is the common name for the JIRA's standard field id. This attribute does not load custom fields.
`{id: 'customfield', format: 'html', params: { fieldId: <field-numeric-id> }}`		HTML representation of a custom field value.

Attribute Spec	Example	Description
`{id: 'project', format: 'id'}`		Project ID for the issues. The `id` format means either a string or a number, depending on what is being used for identifying the object.
`{id: 'editable', format: 'boolean'}`		Boolean value telling whether the item can be edited by the user.

See also CoreAttributeSpecs[860] for examples of bundled attributes.

`getDefaultName()`
Must return default column name, assigned when user adds column of specified type to the structure view. Returns empty string by default.

Example

```
getDefaultName: function() { return 'My Column'; }
```

`isResizable()`
Returns whether the column is resizable or not. Returns `true` by default.

Example

```
isResizable: function() { return false; }
```

`canShrinkWhenNoSpace()`
Returns whether column can shrink beyond minimum size if there's not enough space on the screen. Returns `false` by default.

Example

```
canShrinkWhenNoSpace: function() { return true; }
```

`isAutoSizeAllowed()`
Returns if the column should be auto-resized to fit its contents. Returns `false` by default.

Example

```
isAutoSizeAllowed: function() { return true; }
```

`getMinWidth()`

860 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/CoreAttributeSpecs.html

Returns minimum width of the column in pixels. Returns 27 by default.

Example

```
getMinWidth: function() { return 100; }
```

`getDefaultWidth()`
Returns default width of the column in pixels. Returns 120 by default.

Example

```
getDefaultWidth: function() { return 100; }
```

`getHeaderCellHtml()`
Returns HTML that will be used in the grid header. By default returns cell with column name in default Structure style.

Example

```
getHeaderCellHtml: function() { return '<div>' + this.name + '</div>'; }
```

`getMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column to render the values. See Requesting and Using Metadata(see page 1057) for more information. By default returns `null`, which means that no metadata is needed.

Example

```
getMetadataRequests: function() {
  return {
    status: {
      url: baseUrl + '/rest/api/2/status',
      cacheable: true
    }
  };
}
```

`getSortAttribute()`
Returns attribute specification for sorting when the user clicks on the header. If `null` is returned (the default), the clicking this column header does not result in added sorting transformation.

`isSortDescendingByDefault()`
If returns `true` the initial direction of the sorting will be descending.

ColumnConfigurator Class

window.almworks.structure.api.ColumnConfigurator

ColumnConfigurator class encapculates everything related to column type configuration.

It needs to be subclassed for a particular column type implementation and passed as return value in ColumnType.createConfigurator()(see page 1144) method.

Example

```
var api = window.almworks.structure.api;

var MyColumnConfigurator = api.subClass('MyColumnConfigurator,
api.ColumnConfigurator, {
  getDefaultColumnName: function() { return 'My Column'; }
  getOptions: function() {
    return [ new MyOption1({configurator: this}), new MyOption2({configurator: this})
 ];
  }
});
```

Required Methods

You have to override the following methods in the subclass.

`getColumnTypeName()`
Returns column type name, used in the column configuration panel.

`getDefaultColumnName()`
Returns default column name.

Other Methods

These methods may be optionally overridden.

`init(options)`
Optional initializer.

`getGroupKey()`
Return column preset's group key. See registerColumnGroup()(see page 1133) for reference.

`getMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this configurator to set up the UI. See Requesting and Using Metadata(see page 1057) for more information. By default returns `null`, which means that no metadata is needed.

Example

```
getMetadataRequests: function() {
  return {
    somedata: {                         // metadata key
      url: baseUrl + '/some/data/url', // request URL
      cacheable: true,                  // if the response for this URL can be reused
for other cacheable requests
      extract: function(response) {    // response to the AJAX request
        return response.property || 1; // the actual value for
context.getMetadata('somedata')
      }
    },
    otherdata: {
      url: baseUrl + '/other/data/url',
      cacheable: true
    }
  };
}
```

`getOptions()`
Returns array of column type options. Each option should be a subclass of ColumnOption Class(see page 1141).

Example

```
getOptions: function() {
  return [ new MyOption1({configurator: this}), new MyOption2({configurator: this}) ]
;
}
```

ColumnOption Class

window.almworks.structure.api.ColumnOption

ColumnOption class represents a single column configuration parameter.

It needs to be subclassed for particular column type implementation and passed as return value in ColumnConfigurator.getOptions()(see page 1141) method.

Options are displayed in column configuration dialog one after another with labels on the left and inputs on the right.

Example

```
var api = window.almworks.structure.api;

var MyOption1 = api.subClass('MyOption', api.ColumnOption, {
  title: 'Some option',
  init: function() {
    this.input$ = null;
  },
  createInput: function(div$) {
    this.input$ = div$.append('<input type="text" class="text">').find('input');
    var params = this.spec.params;
    this.input$.on('change', function() {
      if (params.someOptionAvaiable) {
        params.someOption = $(this).val();
        div$.trigger('notify');
      }
    });
  },
  notify: function() {
    var available = this.spec.params.someOptionAvaiable;
    this.input$.val(available ? (this.spec.params.someOption || '42') : '');
    return available;
  }
});
```

Properties

`title`
If set, title is displayed as a label to the left of the input controls. Option title representation may be overridden in #createLabel(div$)(see page 1143) method.

Required Methods

You need to override the following methods.

`createInput(div$)`
Should be overridden to provide custom HTML for the option input. `div$` parameter provides parent option element to append your view to. Created input should trigger `'notify'` event on `div$` to notify Structure of any column parameters change.

Please honor the AUI Forms HTML layout when creating your input controls!

Example

```
createInput: function(div$) {
  var self = this;
  this.input$ = $('<input type="text" class="text">').appendTo(div$).on('change',
function() {
    if (self.spec.params.myOption !== $(this).val()) {
      self.spec.params.myOption = $(this).val();
      div$.trigger('notify');
    }
  });
}
```

Other Methods

`init(options)`
Optional initializer.

`createLabel(div$)`
May be overridden to provide custom HTML view for the input label. `div$` parameter provides parent option element to append your view to. By default creates a right-aligned label with text of the #title(see page 1142) property.

Please honor the AUI Forms HTML layout if you override this method!

`notify()`
This method is called when the column configuration has changed. The implementation may want to update its controls to reflect those changes. The method should return a `boolean` indicating whether this option is available. Unavailable options will not be shown on the configuration panel. The default implementation does nothing and always returns `true`.

Example

```
notify: function() {
  this.input$.val(this.spec.params.myOption);
  return true;
}
```

`isInputValid()`
Returns `true` if the current column specification is valid from the point of view of this option. The column configuration won't be saved unless all of the options approve the specification. The default implementation does nothing and returns `true`.

Example

```
isInputValid: function() {
  // Check that the "field" specification parameter is present.
  return !!this.spec.params.field;
}
```

ColumnType Class

window.almworks.structure.api.ColumnType

ColumnType class represents column type.

It needs to be subclassed for particular column type implementation.

Example

```
var api = window.almworks.structure.api;

var AwesomeColumnType = api.subClass('AwesomeColumnType', api.ColumnType, {
  createSwitchTypePreset: function(context) { return { key:
'com.acme.structure.awesome-column', params: {} }; },
  createAddColumnPresets: function(context) { return [
   { key: 'com.acme.structure.awesome-column', params: {} },
   { key: 'com.acme.structure.awesome-column', name: 'Awesome Column with a Twist',
params: { twist: true } }
  ]; },
  createConfigurator: function(context, spec) { return new
 AwesomeColumnConfigurator({context: context, spec: spec}); },
  createColumn: function(context, spec) { return new AwesomeColumn({context: context,
spec: spec}); }
});

api.registerColumnType(new AwesomeColumnType(), 'com.acme.structure.awesome-column');
```

Methods

`createSwitchTypePreset(context)`
Returns default column specification to use when the user switches to this column type from another column type in the column configuration panel. May return `null` if the column type is unavailable.

`createAddColumnPresets(context)`
Returns an array of column presets (specifications) for this type to be offered to the user in the Add Column panel. May return an empty array if the column type is unavailable.

`createColumn(context, spec)`
Returns a new instance of `Column` subclass for the specified column specification. May return `null` if the specification is invalid, the column type is unavailable, etc.

`createConfigurator(context, spec)`
Returns a new instance of `ColumnConfigurator` subclass for the specified column specification. May return `null` if the specification is invalid, the column type is unavailable, etc.

`getPresetMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column type to create presets. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createSwitchTypePreset()` or `createAddColumnPresets()` is called. See Requesting and Using Metadata(see page 1057) for more information. By default returns `null`, which means that no metadata is needed.

`getColumnMetadataRequests()`

Returns a JavaScript object specifying the metadata needed by this column type to create `Column` instances. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createColumn()` is called, and also will be available to the created `Column` instance via `this.context`. See Requesting and Using Metadata(see page 1057) for more information. By default returns `null`, which means that no metadata is needed.

`getConfigMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column type to create `ColumnConfigurator` instances. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createConfigurator()` is called, and also will be available to the created `ColumnConfigurator` instance via `this.context`. See Requesting and Using Metadata(see page 1057) for more information. By default returns `null`, which means that no metadata is needed.

`getMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column type. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createSwitchTypePreset()`, `createAddColumnPresets()`, `createColumn()`, or `createConfigurator()` is called, and also will be available to the created `Column` and `ColumnConfigurator` instances via `this.context`. See Requesting and Using Metadata(see page 1057) for more information. By default returns `null`, which means that no metadata is needed.

Example

```
getMetadataRequests: function() {
  return {
    somedata: {                         // metadata key
      url: baseUrl + '/some/data/url', // request URL
      cacheable: true,                  // if the response for this URL can be reused
for other cacheable requests
      extract: function(response) {    // response to the AJAX request
        return response.property || 1; // the actual value for
context.getMetadata('somedata')
      }
    },
    otherdata: {
      url: baseUrl + '/other/data/url',
      cacheable: true
    }
  };
}
```

ItemDetailsProvider Class

window.almworks.structure.api.ItemDetailsProvider

ItemDetailsProvider class is the extension point for item details.

To use it, create an ItemDetailsProvider subclass via subClass()[861] function, define necessary properties and then register provider subclass for the specific item type using registerItemDetailsProvider()[862].

---

[861] https://wiki.almworks.com/display/structure/JavaScript+API+Functions#JavaScriptAPIFunctions-window.almworks.structure.api.subClass(className,superclass,prototype)
[862] https://wiki.almworks.com/display/structure2gmaster/JavaScript+API+Functions#JavaScriptAPIFunctions-window.almworks.structure.api.registerItemDetailsProvider(itemType,ProviderClass)

Example

```
var api = window.almworks.structure.api;
var AttachmentDetails = api.subClass('AttachmentDetails', api.ItemDetailsProvider, {
 init: function () {
   ...
 },
 viewport: viewportElement,
 showDetails: function (rowData, life) {
   ...
 },
 ...
});

api.registerItemDetailsProvider('com.acme.my-plugin:type-attachment',
AttachmentDetails);
```

Each ItemDetailsProvider subclass has access to ItemDetailsBridge[863] instance via inherited `'itemDetailsBridge'` property. This object provides additional api for interaction with the Structure app and item details lifecycle.

Properties

`viewport`
Required property. DOM element with the viewport container.

This element should be detached from DOM, Structure itself attaches it to the right place.

`timeoutMessageKey`
Part of i18n key in `'s.itemDetails.stub.title.+$timeoutMessageKey+'` and `'s.itemDetails.stub.body.+$timeoutMessageKey+'` to be used when current row data are loaded for too long.

See extendFocusedRowData()[864]. Has `'timeout'` value by default.

`panelClass`
Class to be set on the outer item details panel div element. Has empty string value by default.

`focusElementClass`
Class of element on the viewport that should be focused when Structure needs to switch focus on the item details panel. Viewport element will be focused if element with this class can't be found or is invisible.

Default value is `'detailsFocusElement'`.

`refreshOnStructureUpdate`
Boolean property. Indicates whether details content must be refreshed via refreshDetails()[865] on Structure update.

Default `false` value means that details will be refreshed only after successful item editing on Structure panel.

---

863 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsBridge+Object
864 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
865 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-refreshDetails(rowData,life)

Methods

`init()`
Initializer that is called during subclass instance creation.

`extendFocusedRowData(rowData)`
Returns a jQuery.Deferred[866] that must either: resolve with `rowData`, possibly extended with additional data, or reject with two parameters:

- `'reason'`: *String* - is used as a i18n key in `'s.itemDetails.stub.title.+$reason+'` and `'s.itemDetails.stub.body.+$reason+'`,
- `'isError'`: *Boolean* - if details should display reason message decorated as error (optional, default: `false`).

If the returned promise doesn't resolve or reject in a timely manner, Structure rejects it with argument equal to provider property timeoutMessageKey[867]. If another row is focused before this promise is done, Structure rejects it automatically.

Default implementation returns a resolved deferred object with `rowData`.

Parameters

`rowData`	*Object*	Holds current row data: • `'rowId'`: *Number* • `'itemId'`: *String*

`showDetails(rowData, life)`
Must be implemented in subclass. The main method - displays the details for the current row in the viewport.

Returns jQuery.Deferred[868] that must resolve when the details are fully shown and the user can interact with them, or reject if the details cannot be shown, with the following parameters:

- `'reason'`: *String* - is used as a i18n key in `'s.itemDetails.stub.title.+$reason+'` and `'s.itemDetails.stub.body.+$reason+'`,
- `'isError'`: *Boolean* - if details should display reason message decorated as error (optional, default: `false`).

If another row is focused while details are being loaded, Structure rejects it.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[869]

866 https://api.jquery.com/category/deferred-object/
867 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-timeoutMessageKey
868 https://api.jquery.com/category/deferred-object/
869 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)

`life`	*Object*	Lifespan for showing details - when it finishes, we no longer need to show the details. Function that may cancel details loading or do some cleanup on lifespan finish can be registered via `life.addDetach(function)`

`beforeHide(rowData)`
Called just before the panel is hidden to perform necessary cleanup.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[870]

`hasUnfinishedEdits()`
Called before hiding details or switching it to another item to check if details has unfinished edits. Confirm dialog appears before details hiding if method returns `true`.

Should be used to prevent unexpected details hiding during content editing.

`getHeaderTitle(rowData)`
Returns text (i18nized) to set to the panel title in the details panel header.

If this method is not overridden or returns a falsy value, a title value is set to `'Item Details'`.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[871]

`getIcon(rowData)`
Returns icon to be displayed in the details header. Return value can be either html string or promise that must be resolved with a html string if it requires asynchronous loading.

The default implementation loads icon html attribute for the focused row.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[872]

`applyWidth(width)`
Updates the displayed details content so it fits the given width. Called on changing details panel width.

Parameters

`width`	*Number*	New panel width

`onDetailsFocusing()`

---

870 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
871 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
872 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)

Called after Structure has switched focus to the details panel to perform additional operations. As an option can be used for focusing iframe document if details content is rendered inside it.

`refreshDetails(rowData, life)`
Performs refresh of details panel content. Called after successful focused item editing on the Structure panel. Called on Structure update only if refreshOnStructureUpdate[873] is set to `true`.

Returns a jQuery.Deferred[874] that must resolve when item details has been refreshed.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[875]
`life`	*Object*	Lifespan for showing details - when it finishes, we no longer need to show details. Function that may cancel details loading or do some cleanup on lifespan finish can be registered via `life.addDetach(function)`

`setDebug(debug)`
Controls additional debug information in logs.

Parameters

`debug`	*Boolean*	Enable/disable debug logging

10.6.5.3 JavaScript API Objects

This page lists objects exposed by the Structure API.

- ItemDetailsBridge Object(see page 1149) — ItemDetailsBridge provides api for interaction with the Structure app and item details lifecycle.

ItemDetailsBridge Object

ItemDetailsBridge provides api for interaction with the Structure app and item details lifecycle.

Item details lifecycle: enabled (panel was open) → visible (ready for interaction) → disabled (panel was closed) → detached (ItemDetails module deactivated).

ItemDetailsBridge instance is accessible in ItemDetailsProvider[876] subclasses via `'itemDetailsBridge'` property.

Properties

`panel$`
jQuery object that contains item details panel element. All item details viewports[877] are appended to this container.

873 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-refreshOnStructureUpdate
874 https://api.jquery.com/category/deferred-object/
875 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
876 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class
877 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-viewport

Methods

`getCurrentRowData()`
Returns `rowData` as returned by corresponding ItemDetailsProvider.extendFocusedRowData()[878] if the current row has the same item type as for which provider is registered, or `null` if it has another type.

`onCurrentRowChange(listener)`
Bind listener to execute each time on changing focused row in Structure.

Parameters

`listener`	*Function(Object)*	function that accepts `rowData` as returned by ItemDetailsProvider.extendFocusedRowData()[879]

`toggleEnabled(enabled)`
Enable/disable details panel layout.

Parameters

`enabled`	*Boolean*	`true` if panel should be enabled and `false` otherwise

`onEnableToggling(listener)`
Bind listener to execute each time the panel layout has been enabled or disabled.

Parameters

`listener`	*Function(Boolean)*	function that accepts `enabled` flag

`isEnabled()`
Returns `true` if panel layout is enabled and `false` otherwise.

`whenVisible(listener)`
Bind listener to execute when details panel becomes ready for interaction.

Parameters

`listener`	*Function*	function to execute

`isVisible()`
Returns `true` if user can interact with the details panel at the moment and `false` otherwise.

`onDetachment(listener)`
Bind listener to perform cleanup on `ItemDetails` module deactivation.

Parameters

`listener`	*Function*	function that performs cleanup

`notifyDetailsEditStarted()`
Should be called when user starts item details content editing. This prevents details refreshing after possible structure updates.

---

878 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
879 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)

`notifyDetailsEditStopped()`
Should be called when user has finished item details content editing. This allows the application to perform details refreshes after structure updates. See `notifyDetailsEditStarted()`.

`notifyDetailsUpdated()`
Should be called when user has finished details update. Needed to notify Structure about possible updates to refresh corresponding rows.

`notifyFocusChanged(focused)`
Should be called when details panel has been focused or lost its focus.

Parameters

`enabled`	*Boolean*	`true` if the panel got focus and `false` otherwise

`shouldBlockActions()`
Returns `true` if item details actions should be blocked at the moment.

`requestRefreshDetails(deferred)`
Request item details refresh. Details panel will be ready for interaction after refresh when corresponding row is updated in Structure. Can be used for Structure and details updates synchronization.

Parameters

`deferred`	*Object*	jQuery.Deferred[880] that must resolve when details panel has been refreshed and Structure row has been updated

`isFocused()`
Returns `true` if details panel is currently focused.

`setFocused(focused)`
Called to make details panel focused/unfocused.

Parameters

`focused`	*Boolean*	`true` if the panel should be focused and `false` otherwise

`focusStealerStarts(stealerId, isFocusedNow)`
Details panel focus can be temporarily taken by some UI elements or dialogs and should be returned back after its closing (or it can be some other circumstances). To provide focus retrieval this function should be called when stealer takes focus from details panel. `focusStealerFinishes()` should be called with the same `stealerId` to return focus back.

Parameters

`stealerId`	*String*	identifier of entity that stole focus from details panel
`focused`	*Boolean*	indicates that details panel should be considered as focused even if it is not so at the moment of function call

`focusStealerFinishes(stealerId, preventFocusing)`
This function should be called when UI element or dialog that stole focus should return it back (i.e. dialog has been closed or disappeared). See `focusStealerStarts()`. Details panel will be focused back when all stealers release their focus.

880 https://api.jquery.com/category/deferred-object/

Parameters

`stealerId`	*String*	identifier of entity that released focus
`preventFocusing`	*Boolean*	indicates that details panel should not be focused even if all focus stealers were released

10.6.6 Web Resource Contexts

The resources from the following web resource contexts are included by Structure pages:

Web resource context	Pages that include it
`structure.widget`	Structure Board(see page 88), Structure Gadgets(see page 565) See Loading Additional Web Resources For Structure Widget(see page 1065) for the recommended way of extending the widget.
`structure.printable`	Printable page(see page 561)

For details about how to use web resource contexts, see Atlassian Developer Documentation[881].

10.7 API Usage Samples

Use the sample plugins to learn by example. Download the source bundle from this page and use it with the latest API version.

10.7.1 Download

File	Modified
user-item-details-1.0.1.jar[882]	Jun 03, 2020 by Julia Atlygina[883]
status-bar-column-3.0.0.jar[884]	Jun 03, 2020 by Julia Atlygina[885]
labels-extender-1.0.3.jar[886]	Jun 03, 2020 by Julia Atlygina[887]

881 https://developer.atlassian.com/display/JIRADEV/Web+Resource+Plugin+Module#WebResourcePluginModule-WebResourceContexts
882 https://wiki.almworks.com/download/attachments/32223506/user-item-details-1.0.1.jar?api=v2
883 https://wiki.almworks.com/display/~atlygina
884 https://wiki.almworks.com/download/attachments/32223506/status-bar-column-3.0.0.jar?api=v2
885 https://wiki.almworks.com/display/~atlygina
886 https://wiki.almworks.com/download/attachments/32223506/labels-extender-1.0.3.jar?api=v2
887 https://wiki.almworks.com/display/~atlygina

File	Modified
custom-itemtype-2.0.0.jar[888]	Jun 03, 2020 by Julia Atlygina[889]
structure-api-examples-6.0.0.zip[890]	Jun 03, 2020 by Julia Atlygina[891]

The provided code is not production-quality and not supported. It is provided as a sample of how one can use Structure API.

The sample code is in public domain – feel free to copy, modify and base your work on it.

10.7.2 Example List

Sample Plugin	Description
simple-plugin	Very basic demo of using `StructureManager`.
status-bar-column	Adds a column to the Structure widget that shows a colored bar, depending on the statuses of the sub-issues.
labels-extender	A plugin that adds Labels Extender, which includes issues in the structure based on issue key mentioned in Labels field of the parent issue.
custom-itemtype	A plugin that adds a new item type based on Jira projects and an inserter which adds projects from one or more categories.
user-item-details	A plugin that adds item details panel support for Jira users.

10.8 Archived Documentation

- Structure 3 API Changes(see page 1154)
- Creating a New Synchronizer(see page 1160)
- structure-synchronizer(see page 1161)

888 https://wiki.almworks.com/download/attachments/32223506/custom-itemtype-2.0.0.jar?api=v2
889 https://wiki.almworks.com/display/~atlygina
890 https://wiki.almworks.com/download/attachments/32223506/structure-api-examples-6.0.0.zip?api=v2
891 https://wiki.almworks.com/display/~atlygina

10.8.1 Structure 3 API Changes

10.8.1.1 State of the API

In Structure 3 we had to change API in an incompatible way because the underlying architecture of the product had changed. If you have integration with Structure 2, most likely it won't work with the new Structure and some effort is needed to migrate the code.

As of Structure versions 3.0 – 3.1, the new API is not yet finalized and thorough documentation is not yet published. We plan to spend additional effort on making the APIs simple, stable and well-documented and publish the final documentation then.

Until that time, it's possible to use the current non-published API with Structure 3, however:

- There's no public documentation on it. The sources of the API artifact are published, but they mostly don't have javadocs yet.
- There will be backwards-incompatible changes while we finalize the API. The concepts and interfaces will stay mostly the same, but some classes may be moved and optimized. This is less likely to impact REST API, although we plan to introduce new REST APIs that would be simpler than the low-level API we have now.
- There will be new interfaces that would make it easier to deal with the new concepts. Right now it may be a little "low-level" and somewhat more complicated than it needs to be.

Although the documentation about the new API is not available, Structure team will be happy to assist you in migrating your code to work with Structure 3.

This article lists some of the most frequently used API calls. If you need to do something that is not covered by this article or have any questions, please write us at Tempo Support[892].

10.8.1.2 Conceptual Changes

Forests and Rows

In Structure 2, a structure's content was called a ***forest*** . That is still the case, however, the forest now contains ***rows*** rather than issues. A row has a Row ID – a long integer primary key for the row. Given row ID, you can retrieve information about the item displayed in that row.

The data structure that represents a forest didn't change. Previously a forest was represented by an array of pairs `(issueID, depth)`. Now the forest is represented by an array of pairs `(rowID, depth)`.

The concept of a row may seem superfluous, but it's actually required for uniquely identifying a specific position in a forest. Issue ID (or Item ID) is not sufficient because an issue can be located at multiple places in the forest.

Items

Each row has an associated ***item*** – an issue, a folder, a project or any other type of items. In Structure 3, item types are extendable and an add-on may provide additional types of items to Structure. An item is identified by ***Item Identity*** , which consists of:

892 https://tempo-io.atlassian.net/servicedesk/customer/portal/6/group/1051/create/45?customfield_12525=12553&customfield_12526=12514

- Item Type, represented by a complete module key of a JIRA plugin module that provides the item type, and
- Item ID, represented by *either* a long integer (for example, issue ID for issues) or by a string (a user key for users).

Sometimes item type is omitted; in that case the implied item type is "issue".

Some of the most popular item types are:

Item Type	Module Key	Meaning of Item ID	Comments
Issue	`com.almworks.jira.structure:type-issue`	Issue ID (long)	Default when item type is not specified
Folder	`com.almworks.jira.structure:type-folder`	Either folder ID (long) or folder i18n name (string)	Folders are introduced in Structure 3
User	`com.almworks.jira.structure:type-user`	User key (string)	
Generator	`com.almworks.jira.structure:type-generator`	Generator ID (long)	A generator is an automation rule embedded in the structure.
Page	`com.almworks.structure.pages:type-confluence-page`	Page ID (ID modulo 1e9)	Confluence pages are added as a type by Structure.Pages extension

Attributes

Attributes are a generalization of a JIRA's issue fields. An attribute is something that can be calculated for an item. For example, an issue has such attributes as "summary", "key", "priority". But purely Structure-related values are also attributes, such as "sequential number", or "aggregate progress", or "sum of story points". The attributes can also be retrieved for any types of items – for a Confluence page (provided by Structure.Pages extension), "summary" would be the title of the page, "labels" would be the labels, and a new attribute "author" would provide the initial author of the page.

Attributes are identified by an attribute specification, or ***attribute* spec**. It is usually represented as a JSON object with an ID and parameters.

Concept Comparison

	Structure 2	Structure 3
A structure's content is ...	Forest	Forest
Things that can be placed into a structure are ...	Issues	Items (including issues)
Forest consists of ...	Issues	Rows
A position in a forest is identified by ...	Issue ID	Row ID

	Structure 2	Structure 3
A value in the Structure grid is displayed by ...	Column	Column
A column requests from the server ...	Fields	Attributes

10.8.1.3 REST API

Retrieving Structure Forest

```
GET /rest/structure/2.0/forest/latest?s={%22structureId%22:$id}
```

This method retrieves a content of a structure. If structure has generators, the generated content is returned. Generators are preserved in the forest.

Parameters:

- `$id` – structure ID

Return value sample:

```
{
  "spec":{"structureId":171},
  "formula":"10394:0:4/356,10332:0:14707,10374:1:5/240,10348:2:14717",
  "itemTypes":{
    "4":"com.almworks.jira.structure:type-generator",
    "5":"com.almworks.jira.structure:type-folder"
  },
  "version":{
    "signature":-1659607419,
    "version":1
  }
}
```

In this reply, the most important part is "formula", which contains serialized information about the forest, much like in Structure 2.

Each component (delimited by comma) represents a row and looks like this: `10374:1:5/240`. In this example, the numbers are:

- `10374` is the row ID,
- `1` is the row depth,
- `5/240` is the item identity. If the row contains an issue, it's just issue ID, otherwise it has the format of `<item type>/<long item id>`, or `<item type>//<string item id>`. Item type is a number, which is expanded in the "itemTypes" map in the reply.

Updating a Structure Forest

```
POST /rest/structure/2.0/forest/update
```

Parameters:

```
{
  "spec": { "structureId": <id> }, // use structure ID
  "version": { "signature": <signature>, "version": <version> },  // use last seen signature and version
  "actions": [
     {
         "action": "add",
         "under": 0,        // at the top level
         "after": 123,      // after row ID 123 (not issue iD!)
         "before": 456,   // before row ID 456
         "forest": "-100:0:10001"   // insert issue 10001, -100 is the temporary row ID which will be mapped into the real row ID when the method returns
     },
     {
         "action": "move", // works like previously, only row IDs instead of issue IDs
         "rowId": 123,
         "under": 456,
         "after": 0,
         "before": 124
     },
    {
       "action": "remove",
       "rowId": 442
     }
  ]
}
```

Creating a structure

```
POST /rest/plugins/structure/2.0/structure
```

Parameters:

```
{
   "name": "my structure",
   "description": "my description",
   "permissions": [   ]    // same format you see when you GET structure
}
```

Deleting a structure

```
DELETE /rest/plugins/structure/2.0/structure/<id>
```

10.8.1.4 Java API

Versions

As Structure 3 API is finalized, it's getting a lot of refactoring and version changes. A new Structure version may have a backward-incompatible API, although incompatibilities may be isolated and your code has a good chance to work fine. However, the major version is promoted every time a backward-incompatible change is made, therefore you need to carefully set up the version of imported API packages – either set them optimistically (for example, `[12,15)` – up to version 15) and test your integration with a new release to see that there are no errors; or set the version as usual – for example, `[12,13)` – but then you might need to recompile with each new release of the API. The latter approach is recommended for in-house customizations.

Version	Supported JIRA Versions	Introduced in Structure Version	OSGi Import	OSGI Import (Optimistic)
12.0.0	JIRA 6.3+	3.0.0	"[12,13)"	"[12,15)"
12.1.0	JIRA 6.3+	3.0.1	"[12.1,13)"	"[12.1,15)"
13.0.0	JIRA 6.3+	3.1.0	"[13,14)"	"[13,16)"
13.0.1	JIRA 6.3+	3.1.1	"[13,14)"	" [13,16)"

The API versions and sources are available from the public Maven repositories – http://mvnrepository.com/artifact/com.almworks.jira.structure/structure-api

Retrieving Structure's Forest

To get the content of a structure, you need to use `ForestService` interface, which can be injected. It has `getForestSource()` method that will return a `ForestSource` given `ForestSpec`, which is a specification of what kind of forest you are retrieving. For getting just a content of a structure, use `ForestSpec.structure(structureId)`. Once you have a `ForestSource`, you can use `forestSource.getLatest().getForest()` to retrieve an instance of `Forest` – which should be familiar from the Structure 2 API.

But now `Forest` contains row IDs, not issue IDs, so to get information about what issues (or other items) are in the forest, you need to "dereference" each row ID.

Working with Rows

For working with rows, use `RowManager`. To get an item ID from a row ID, use `rowManager.getRow(rowId).getItemId()`. This gives you `ItemIdentity` instance. To see if it is an issue, use `CoreIdentities.isIssue(itemId)` and to get issue ID in that case, use `itemId.getLongId()`.

To get all row IDs for a given issue ID (for example, to find an issue in a forest), you can use `rowManager.findRows(CoreIdentities.issue(issueId))`. These row IDs may be from multiple forests, so you need to see if the forest that you have contains some of those IDs.

Getting Totals and Other Values

To calculate totals or other Structure-calculated values, you need to use `StructureAttributeService`.

`StructureAttributeService.getAttributeValues()` has the following parameters:

- `ForestSpec` – use the same forest spec that you use to retrieve the forest;
- row IDs – you need to specify for which rows (not issues!) the values are requested;
- a collection of `AttributeSpec` – specify which attributes are requested.

You need to build a list of attribute specs to specify what to calculate. There are several ways to get a correct attribute spec:

- Some specs are defined in `CoreAttributeSpecs`.
- You can build a spec using `AttributeSpecBuilder`.
- You can parse a JSON representation of a spec into a `Map`, then extract "id" and "params".

Examples:

Attribute	Spec
Story Points	`AttributeSpecBuilder .create(“customfield", ValueFormat.NUMBER) .params() .set("fieldId", 10000) // 10000 - the id of "Story Points" custom field .build()`
∑ Story Points	`AttributeSpecBuilder .create("sum", ValueFormat.NUMBER) .params() .setAttribute(storyPoints) // storyPoints = the attribute spec for Story Points .set("distinct", true) // exclude duplicates .build()`

Changing Structure

To change a structure, you need to use `UpdatableForestSource.apply()` method. Each update is a separate transaction – the concept of a ForestTransaction used in Structure 2 has been removed.

To get an instance of `UpdatableForestSource` you need to cast `ForestSource` retrieved from `ForestService`.

Examples:

Operation	Code
Add an issue with ID 10200 to structure, under parent row with ID 1040, after row with ID 1900 and before row with ID 2000	`forestSource.apply(new UpdatableForestSource.Update.Add( CoreIdentities.issue(10200), 1040, 1900, 2000))`
Remove rows with IDs 10100 and 10102	`forestSource.apply(new UpdatableForestSource.Update.Remove( LongArray.create(10100, 10102)))`
Move row with ID 1010 as the first row under parent row with ID 1040, before a row with ID 1060	`forestSource.apply(new UpdatableForestSource.Update.Move( LongArray.create(1010), 1040, 0, 1060))`

10.8.2 Creating a New Synchronizer

> Synchronizers is an older, outdated mechanism for automation of Structure and Jira changes. It is being replaced with Generators and Effectors.
>
> If you'd like to create your own automation, we strongly suggest using Generators and/or Effectors.

Structure comes with a number of bundled synchronizers(see page 983), but you can add another synchronizer to the system, allowing Structure users to install it on structures and run export / import.

10.8.2.1 Implement `StructureSynchronizer`

Create your implementation of StructureSynchronizer[893] interface. Use AbstractSynchronizer[894] as the base class.

10.8.2.2 Define `structure-synchronizer` Module

Add structure-synchronizer(see page 1161) module to your `atlassian-plugin.xml`, referring to your implementation of the `StructureSynchronizer`.

10.8.2.3 Test Thoroughly

Test how your synchronizer works when other synchronizers are also installed onto the same structure.

10.8.2.4 Sample Project

This project can be used to bootstrap writing your own synchronizer. It compiles into a working plugin, which does not do anything except writing to console at the times the synchronizer would do some work.

You can download the sources zip with the sample synchronizers at API Usage Samples(see page 1152) page.

10.8.3 structure-synchronizer

Synchronizer module allows you to plug additional synchronizers into Structure.

10.8.3.1 Module description sample

Here's a template of a synchronizer module declaration, and explanation of the parameters follows.

```
  <structure-synchronizer key="module-key" order="100"
                          class="com.company.your.plugin.sync.SyncClass">
    <label key="label.i18n.key">Name of Synchronizer</label>
    <description key="description.i18n.key">Description of Synchronizer</description>
    <rules key="rules.i18n.key">Large text to be shown at the top of synchronizer's
configuration page.</rules>
    <resource type="velocity" name="form" location="/templates/myplugin/sync-form.vm"
/>
  </structure-synchronizer>
```

Element	Required	Description
`structure-synchronizer`	Yes	The module descriptor.

893 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/StructureSynchronizer.html
894 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/AbstractSynchronizer.html

Element	Required	Description
`@key`	Yes	Unique module key within the plugin.
`@order`	Yes	Order of the synchronizer among other synchronizers, whenever a list of synchronizers is present.
`@class`	Yes	The class that implements the synchronizer. Must implement StructureSynchronizer[895]. It is recommended to extend AbstractSynchronizer[896].
`label`	Yes	The name of the synchronizer.
`description`	No	Description of the synchronizer.
`rules`	No	The text that is shown at the top of the synchronizer configuration page. Could be a large text.
`resource[@name="form"]`	Yes	A velocity template that contains the form for the synchronizer parameters.

10.9 Open Source Licenses

Structure for Jira is made possible by open source software.

The following is a list of open source libraries used in the product and links to their respective license agreements.

Component / Library	License
Annotations (JetBrains)[897]	Apache 2.0[898]
Apache Commons[899]	Apache 2.0[900]

895 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/StructureSynchronizer.html
896 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/AbstractSynchronizer.html
897 https://github.com/JetBrains/java-annotations
898 https://www.apache.org/licenses/LICENSE-2.0
899 https://commons.apache.org/
900 https://www.apache.org/licenses/LICENSE-2.0

Component / Library	License
Apache Derby[901]	Apache 2.0[902]
Apache HttpClient[903]	Apache 2.0[904]
Apache POI[905]	Apache 2.0[906]
HPPC[907]	Apache 2.0[908]
Jackson[909]	Apache 2.0[910]
JavaEWAH[911]	Apache 2.0[912]
Javolution[913]	BSD license[914]
Jsoup[915]	MIT license[916]
jQuery[917]	MIT license[918]
jQuery UI[919]	MIT license[920]
hoverIntent[921]	MIT license[922]
Moment.js[923]	MIT license[924]
Font Awesome[925]	SIL OFL 1.1, MIT license[926]

901 https://db.apache.org/derby/
902 https://www.apache.org/licenses/LICENSE-2.0
903 https://hc.apache.org/httpcomponents-client-ga/
904 https://www.apache.org/licenses/LICENSE-2.0
905 https://poi.apache.org/
906 https://www.apache.org/licenses/LICENSE-2.0
907 https://labs.carrotsearch.com/hppc.html
908 https://www.apache.org/licenses/LICENSE-2.0
909 https://github.com/FasterXML/jackson
910 https://www.apache.org/licenses/LICENSE-2.0
911 https://github.com/lemire/javaewah
912 https://www.apache.org/licenses/LICENSE-2.0
913 http://javolution.org/
914 http://javolution.org/LICENSE.txt
915 https://jsoup.org/
916 https://jsoup.org/license
917 https://jquery.com/
918 https://jquery.org/license/
919 https://jqueryui.com/
920 https://jquery.org/license/
921 https://github.com/briancherne/jquery-hoverIntent
922 https://opensource.org/licenses/MIT
923 https://momentjs.com/
924 https://github.com/moment/moment/blob/develop/LICENSE
925 https://fontawesome.com/
926 https://fontawesome.com/v4.7.0/license/

11 Download

The easiest way to install Structure and its add-ons is by using the Jira add-on manager. They can also be installed manually using the downloads below.

11.1 Download Structure Plugin

Please select the version appropriate for your Jira.

Structure version	Jira version	Download link	Publish Date	End of Life	md5sum
8.3.1	Jira 8.20+	structure-8.3.1.jar[927]	2023-05-23	2024-05-23	ba71dd4b1531af8180c59095149155bb

11.2 Download Structure Extensions

Extension	Version	Compatibility	Download link	Publish Date	End of Life
Structure.Testy	2.8.1	Jira 8.13+ Structure 7.0+	testy-2.8.1.jar[928] md5 b5c6eef13cfbc3c38606fa5f95d8b486	2022-12-28	2023-12-28
Structure.Pages Confluence Helper for Structure.Pages	See Structure.Pages Download[929]				

End-User License Agreement (PDF)[930]

927 https://d1.almworks.com/.files/structure-8.3.1.jar
928 https://d1.almworks.com/.files/testy-2.8.1.jar
929 https://wiki.almworks.com/display/pages/Download
930 http://almworks.com/EULA-Structure.pdf

End of Life:

- For beta and release candidate versions: Beta versions may be scheduled to expire after a certain amount of time. In any case, the lifespan of such a pre-release version is not supposed to be longer than 3 months. You will need to upgrade to a newer version before the end-of-life date.
- For other versions: You will be able to use this version of the plugin indefinitely; however, after the end-of-life date, the support for the version is limited.

11.3 What's Next?

See Structure Release Notes (see page 681)and Getting Started with Structure(see page 56)

11.4 Documentation

When opened in a viewport, the user will be redirected to: https://wiki.almworks.com/documentation/structure.

Download documentation for Structure:

File	Modified
Structure-Documentation-v7_3.xml.zip[931]	Oct 11, 2022 by Jaramy Conners[932]
Structure-Documentation-v7_3.pdf[933]	Oct 11, 2022 by Jaramy Conners[934]

For previous versions of Structure, see Version Index[935].

11.5 Download Archive

This page contains links to older version of the Structure plugin.

931 https://wiki.almworks.com/download/attachments/155812178/Structure-Documentation-v7_3.xml.zip?api=v2
932 https://wiki.almworks.com/display/~jaramy
933 https://wiki.almworks.com/download/attachments/155812178/Structure-Documentation-v7_3.pdf?api=v2
934 https://wiki.almworks.com/display/~jaramy
935 https://wiki.almworks.com/display/docs/Structure+Documentation

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
8.3.1	Jira 8.20+	structure-8.3.1.jar[936]	2023-05-23	2024-05-23	ba71dd4b1531af8180c59095149155bb
8.3	Jira 8.20+	structure-8.3.jar[937]	2023-04-25	2024-04-25	40b6e1c1867a5f2198f78fac540dc00c
8.2.2	Jira 8.13+	structure-8.2.2.jar[938]	2023-01-26	2024-01-26	78cf3e5e481a63c642e012f630d30cd4
8.2.1	Jira 8.13+	structure-8.2.1.jar[939]	2023-01-09	2024-01-09	1afb2f43a109a6acde4d5ac12e213128
8.2	Jira 8.13+	structure-8.2.jar[940]	2022-12-15	2023-12-15	d93e52bd4c80dc05dcd71357b1aa9302
8.1.2	Jira 8.13+	structure-8.1.2.jar[941]	2022-10-13	2023-10-13	2dd23b6b93c476541bf4b97c251e60d9
8.1.1	Jira 8.13+	structure-8.1.1.jar[942]	2022-08-30	2023-08-30	cdc2fc17b0de2ede78f47b49eb8f7971
8.1	Jira 8.13+	structure-8.1.0.jar[943]	2022-08-22	2023-08-22	df5ac073ba0439bbc6b8160ad9a63c42
8.0.1	Jira 8.13+	structure-8.0.1.jar[944]	2022-04-25	2023-04-25	81af5383b9f02d8286857b8c11f3fb46
8.0	Jira 8.13+	structure-8.0.0.jar[945]	2022-03-30	2023-03-30	52fcbe20dfe4562105abdff5ff527e60

936 https://d1.almworks.com/.files/structure-8.3.1.jar
937 https://d1.almworks.com/.files/structure-8.3.0.jar
938 https://d1.almworks.com/.files/structure-8.2.2.jar
939 https://d1.almworks.com/.files/structure-8.2.1.jar
940 https://d1.almworks.com/.files/structure-8.2.0.jar
941 https://d1.almworks.com/.files/structure-8.1.2.jar
942 https://d1.almworks.com/.files/structure-8.1.1.jar
943 https://d1.almworks.com/.files/structure-8.1.0.jar
944 https://d1.almworks.com/.files/structure-8.0.1.jar
945 https://d1.almworks.com/.files/structure-8.0.0.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
7.4	Jira 8.13+	structure-7.4.0.jar[946]	2021-12-21	2022-12-22	6c88922a6ca8a8bf26a578a6d0bdabfb
7.3	Jira 8.5+	structure-7.3.0.jar[947]	2021-10-22	2022-10-22	4d93d4715a6cb0d04134ed38660777ef
7.2	Jira 8.5+	structure-7.2.0.jar[948]	2021-08-31	2022-08-31	fefaf3782241dcdd2842e47ef61cd176
7.1	Jira 8.5+	structure-7.1.0.jar[949]	2021-07-22	2022-07-22	4d3dbb20f633e1ce2b3ed0e83bd72a68
7.0	Jira 8.5+	structure-7.0.0.jar[950]	2021-06-07	2022-06-07	2372ff56856dc5e6379ca5146796de78
6.6	Jira 8.5+	structure-6.6.0.jar[951]	2021-03-18	2022-03-18	6f78721dd5b279335e97d1ed737bd291
6.5.2	Jira 8.5+	structure-6.5.2.jar[952]	2021-02-22	2022-02-22	d3a1a41cbe9dfefeabe134f7273b8602
6.5.1	Jira 8.5+	structure-6.5.1.jar[953]	2021-02-08	2022-02-08	2eb327cf0cf7cfde1304a7c4211dc2e3
6.5	Jira 8.5+	structure-6.5.0.jar[954]	2021-02-01	2022-02-01	92396a9f8b67d015dbcdd5856d5bc948
6.4	Jira 7.13+	structure-6.4.0.jar[955]	2020-12-07	2021-12-07	187461a726b301bdefd3e86ec5d12375
6.3.2	Jira 7.13+	structure-6.3.2.jar[956]	2020-11-18	2021-11-18	aa16b82b6a981185939d5748ff7d3e73

946 https://d1.almworks.com/.files/structure-7.4.0.jar
947 https://d1.almworks.com/.files/structure-7.3.0.jar
948 https://d1.almworks.com/.files/structure-7.2.0.jar
949 https://d1.almworks.com/.files/structure-7.1.0.jar
950 https://d1.almworks.com/.files/structure-7.0.0.jar
951 https://d1.almworks.com/.files/structure-6.6.0.jar
952 https://d1.almworks.com/.files/structure-6.5.2.jar
953 https://d1.almworks.com/.files/structure-6.5.1.jar
954 https://d1.almworks.com/.files/structure-6.5.0.jar
955 https://d1.almworks.com/.files/structure-6.4.0.jar
956 https://d1.almworks.com/.files/structure-6.3.2.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
6.3.1	Jira 7.13+	structure-6.3.1.jar[957]	2020-10-10	2021-10-10	1c07db50c8b0a4f4f42d1a5066816e50
6.3	Jira 7.13+	structure-6.3.0.jar[958]	2020-10-08	2021-10-08	4f3f450bc18d1196f37696e27223975e
6.2.1	Jira 7.13+	structure-6.2.1.jar[959]	2020-09-02	2021-09-02	a4bdd94aeb6a0e06bfc9eab775b66dae
6.2	Jira 7.13+	structure-6.2.0.jar[960]	2020-08-10	2021-08-10	70b5e695ed8f13bee1e7bed4a5f165c9
6.1.2	Jira 7.13+	structure-6.1.2.jar[961]	2020-07-06	2021-07-06	5b45cc3ace69ff03352576c407892482
6.1.1	Jira 7.13+	structure-6.1.1.jar[962]	2020-06-17	2021-06-17	8acdff591068c6b48711539fd2ffc7b3
6.1	Jira 7.13+	structure-6.1.0.jar[963]	2020-06-03	2021-06-03	b58eac7d8e02a77492907bd95616aaf2
6.0.1	Jira 7.13+	structure-6.0.1.jar[964]	2020-05-05	2021-05-05	0b2a035debd070f9a41135bf2b003bdb
6.0	Jira 7.13+	structure-6.0.0.jar[965]	2020-03-26	2021-03-26	a1ba401d9eb311f41a1ac2c8ef643524
5.6.4	Jira 7.6+	structure-5.6.4.jar[966]	2020-05-05	2021-05-05	33acfaa18df27b3ed98e28e666693e48

957 https://d1.almworks.com/.files/structure-6.3.1.jar
958 https://d1.almworks.com/.files/structure-6.3.0.jar
959 https://d1.almworks.com/.files/structure-6.2.1.jar
960 https://d1.almworks.com/.files/structure-6.2.0.jar
961 https://d1.almworks.com/.files/structure-6.1.2.jar
962 https://d1.almworks.com/.files/structure-6.1.1.jar
963 https://d1.almworks.com/.files/structure-6.1.0.jar
964 https://d1.almworks.com/.files/structure-6.0.1.jar
965 https://d1.almworks.com/.files/structure-6.0.0.jar
966 https://d1.almworks.com/.files/structure-5.6.4.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
5.6.3	Jira 7.6+	structure-5.6.3.jar[967]	2020-01-24	2021-01-24	`f182eb51327d57eefdfcdfa9bdee0a1b`
5.6.2	Jira 7.6+	structure-5.6.2.jar[968]	2019-12-24	2020-12-24	`a77fb1c7ae3edbd343b9ae046393af7e`
5.6.1	Jira 7.6+	structure-5.6.1.jar[969]	2019-10-14	2020-10-14	`d58768e59ed9990c5a9f5cb3fa2a7d0a`
5.6	Jira 7.6+	structure-5.6.0.jar[970]	2019-09-19	2020-09-19	`3c70f95463289ce478e576697205be86`
5.5.1	Jira 7.6+	structure-5.5.1.jar[971]	2019-08-15	2020-08-15	`69cde87aafcec1b1009db14a04efee1a`
5.5	Jira 7.6+	structure-5.5.0.jar[972]	2019-07-18	2020-07-18	`cfdae3d1a6940d0bdd42689ea51e0867`
5.4.1	Jira 7.6+	structure-5.4.1.jar[973]	2019-07-05	2020-07-05	`54ea92f79d656e507a08066d6166966d`
5.4	Jira 7.6+	structure-5.4.0.jar[974]	2019-06-05	2020-06-05	`e5b78ce8995c9b4f4cc666fba316ebb6`
5.3	Jira 7.6+	structure-5.3.0.jar[975]	2019-03-26	2020-03-26	`092763ea41d479e4ed9071ba6892eb6d`

967 https://d1.almworks.com/.files/structure-5.6.3.jar
968 https://d1.almworks.com/.files/structure-5.6.2.jar
969 https://d1.almworks.com/.files/structure-5.6.1.jar
970 https://d1.almworks.com/.files/structure-5.6.0.jar
971 https://d1.almworks.com/.files/structure-5.5.1.jar
972 https://d1.almworks.com/.files/structure-5.5.0.jar
973 https://d1.almworks.com/.files/structure-5.4.1.jar
974 https://d1.almworks.com/.files/structure-5.4.0.jar
975 https://d1.almworks.com/.files/structure-5.3.0.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
5.2.2	Jira 7.6+	structure-5.2.2.jar[976]	2019-03-01	2020-03-01	4e80a6e6546aeaba4c9111edc0bfd83d
5.2.1	Jira 7.6+	structure-5.2.1.jar[977]	2019-02-06	2020-02-06	a435400d6acc22013921f5160d62aa58
5.2	Jira 7.6+	structure-5.2.0.jar[978]	2018-12-27	2019-12-27	9c22f0934b91ae85484aa18ed70870d1
5.1.1	Jira 7.2+	structure-5.1.1.jar[979]	2020-05-05	2021-05-05	bdca2f7b51233164bc9fb5219462a6b3
5.1	Jira 7.2+	structure-5.1.0.jar[980]	2018-10-25	2019-10-25	1e1d7dfd3246b3b88415f2463222ffdc
5.0	Jira 7.2+	structure-5.0.0.jar[981]	2018-08-16	2019-08-16	05a4109afbdea54a7e877ef5140de500
4.6.5	Jira 7.2+	structure-4.6.5.jar[982]	2018-06-13	2019-06-13	a2065e24f6cbd67960698bafd15581db
4.6.3	Jira 7.2+	structure-4.6.3.jar[983]	2018-05-14	2019-05-14	fc99383bd524318f6d14b6f86d408b0b
4.6.1	Jira 7.2+	structure-4.6.1.jar[984]	2018-04-27	2019-04-27	f0fe1fd44015b2cba9024cfba1e8e438
4.6.0	Jira 7.2+	structure-4.6.0.jar[985]	2018-03-27	2019-03-27	e582186b122c48c66ce123da25d37ce9
4.5.1	Jira 7.2+	structure-4.5.1.jar[986]	2018-01-19	2019-01-19	fdb7abfe158dcae24005621ab014fd67

976 https://d1.almworks.com/.files/structure-5.2.2.jar
977 https://d1.almworks.com/.files/structure-5.2.1.jar
978 https://d1.almworks.com/.files/structure-5.2.0.jar
979 https://d1.almworks.com/.files/structure-5.1.1.jar
980 https://d1.almworks.com/.files/structure-5.1.0.jar
981 https://d1.almworks.com/.files/structure-5.0.0.jar
982 https://d1.almworks.com/.files/structure-4.6.5.jar
983 https://d1.almworks.com/.files/structure-4.6.3.jar
984 https://d1.almworks.com/.files/structure-4.6.1.jar
985 https://d1.almworks.com/.files/structure-4.6.0.jar
986 https://d1.almworks.com/.files/structure-4.5.1.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
4.5.0	Jira 7.2+	structure-4.5.0.jar[987]	2017-12-28	2018-12-28	`543ed2f073f85710af33e109f9ac1dea`
4.4.0	Jira 7.2+	structure-4.4.0.jar[988]	2017-11-29	2018-11-29	`df72ef8c8a52257d1b9a3d6085cc0873`
4.3.0	Jira 7.2+	structure-4.3.0.jar[989]	2017-10-20	2018-10-20	`47bc93338711b19250ed98e53eeced25`
4.2.0	Jira 7.2+	structure-4.2.0.jar[990]	2017-08-25	2018-08-25	`56bf4cb46edb400d9e5e0ec71a89161b`
4.1.2	Jira 7.1+	structure-4.1.2.jar[991]	2020-05-05	2021-05-05	`1f338de829e7dfdd3152aca3c15f1c37`
4.1.1	Jira 7.1+	structure-4.1.1.jar[992]	2017-10-20	2018-10-20	`8aa826eeae801fd022252f8274172184`
4.1.0	Jira 7.1+	structure-4.1.0.jar[993]	2017-06-19	2018-06-19	`975594ce5358b2f1d8d5843ce060c790`
4.0.0	Jira 7.1+	structure-4.0.0.jar[994]	2017-04-26	2018-04-26	`4cfed9da9885477d3a16576ab9559057`
3.6.0	Jira 7.1+	structure-3.6.0.jar[995]	2017-04-04	2018-04-04	`0fe8a3e02429840e7331ddebb2f7c802`
3.5.2	Jira 7.0+	structure-3.5.2.jar[996]	2020-05-05	2021-05-05	`43fe189880fa7b97ad551a2b94a2932f`
3.5.1	Jira 7.0+	structure-3.5.1.jar[997]	2017-06-19	2018-06-19	`f7bd5d660a0b66df0decf6b6ac2e8e6c`

987 https://d1.almworks.com/.files/structure-4.5.0.jar
988 https://d1.almworks.com/.files/structure-4.4.0.jar
989 https://d1.almworks.com/.files/structure-4.3.0.jar
990 https://d1.almworks.com/.files/structure-4.2.0.jar
991 https://d1.almworks.com/.files/structure-4.1.2.jar
992 https://d1.almworks.com/.files/structure-4.1.1.jar
993 https://d1.almworks.com/.files/structure-4.1.0.jar
994 https://d1.almworks.com/.files/structure-4.0.0.jar
995 https://d1.almworks.com/.files/structure-3.6.0.jar
996 https://d1.almworks.com/.files/structure-3.5.2.jar
997 https://d1.almworks.com/.files/structure-3.5.1.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
3.5.0	Jira 7.0+	structure-3.5.0.jar[998]	2017-01-30	2018-01-30	8ef314fae4039e9fa3aaca82b872c413
3.4.1	Jira 7.0+	structure-3.4.1.jar[999]	2016-12-30	2017-12-30	54a0bb98afb025fe1ec9c5453316cd55
3.4.0	Jira 7.0+	structure-3.4.0.jar[1000]	2016-12-12	2017-12-12	e5ab3ded92f25e58ea442723a2643c40
3.3.5.jira6	Jira 6.4.x	structure-3.3.5.jira6.jar[1001]	2017-06-19	2018-06-19	d1feb277d8c071296df9e4176195632d
3.3.4.jira6	Jira 6.4.x	structure-3.3.4.jira6.jar[1002]	2017-01-30	2018-01-30	fb12064eb4c81af53cdbe674176cc7c2
3.3.3.jira7	Jira 7.0+	structure-3.3.3.jira7.jar[1003]	2016-11-15	2017-11-15	ebab71cf97a9768db69b39b39e958382
3.3.3.jira6	Jira 6.4.x	structure-3.3.3.jira6.jar[1004]	2016-11-15	2017-11-15	303e275eac1196732a4bb44680f10fe7
3.3.2.jira7	Jira 7.0+	structure-3.3.2.jira7.jar[1005]	2016-10-03	2017-10-03	1dc7fb3bf533c6e47bec6c890376a29e
3.3.2.jira6	Jira 6.4.x	structure-3.3.2.jira6.jar[1006]	2016-10-03	2017-10-03	eea5925cbf738e8fed62be657a9e1f2d
3.3.1.jira7	Jira 7.0+	structure-3.3.1.jira7.jar[1007]	2016-09-22	2017-09-22	e11d7d726b1f78b1d632407d39254014
3.3.1.jira6	Jira 6.4.x	structure-3.3.1.jira6.jar[1008]	2016-09-22	2017-09-22	312e192ab68cc1086388b7bf5f5cc8b1

998 https://d1.almworks.com/.files/structure-3.5.0.jar
999 https://d1.almworks.com/.files/structure-3.4.1.jar
1000 https://d1.almworks.com/.files/structure-3.4.0.jar
1001 https://d1.almworks.com/.files/structure-3.3.5.jira6.jar
1002 https://d1.almworks.com/.files/structure-3.3.4.jira6.jar
1003 https://d1.almworks.com/.files/structure-3.3.3.jira7.jar
1004 https://d1.almworks.com/.files/structure-3.3.3.jira6.jar
1005 https://d1.almworks.com/.files/structure-3.3.2.jira7.jar
1006 https://d1.almworks.com/.files/structure-3.3.2.jira6.jar
1007 https://d1.almworks.com/.files/structure-3.3.1.jira7.jar
1008 https://d1.almworks.com/.files/structure-3.3.1.jira6.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
3.3.0.jira7	Jira 7.0+	structure-3.3.0.jira7.jar[1009]	2016-09-09	2017-09-09	a62f6588ddc5f090724f1dcb5a87f6ae
3.3.0.jira6	Jira 6.4.x	structure-3.3.0.jira6.jar[1010]	2016-09-09	2017-09-09	8cb038673694bcbe27ca07fd736ac8e4
3.2.3.jira6	Jira 6.3-6.4.x	structure-3.2.3.jira6.jar[1011]	2017-06-19	2018-06-19	703ef49c587f688d9989e8b3032a5c87
3.2.2.jira6	Jira 6.3-6.4.x	structure-3.2.2.jira6.jar[1012]	2017-01-30	2018-01-30	f1dab1e8a99ed88114a759e2284e5441
3.2.1.jira7	Jira 7.0+	structure-3.2.1.jira7.jar[1013]	2016-07-05	2017-07-05	25d9e2171a9de36cec603b6a17ec7525
3.2.1.jira6	Jira 6.3 – 6.4.x	structure-3.2.1.jira6.jar[1014]	2016-07-05	2017-07-05	3d4d50d920280c78d6e982011fba8b66
3.2.0.jira7	Jira 7.0+	structure-3.2.0.jira7.jar[1015]	2016-06-18	2017-06-18	c4e63140c4fb86798c05fc7f2c032b36
3.2.0.jira6	Jira 6.3 – 6.4.x	structure-3.2.0.jira6.jar[1016]	2016-06-18	2017-06-18	2d9d4f42a7e55e312e9e12ebf0f3af5f
3.1.1.jira7	Jira 7.0+	structure-3.1.1.jira7.jar[1017]	2016-06-02	2017-06-02	cd355ec9ff95e335f9e7d587e48335dd
3.1.1.jira6	Jira 6.3 – 6.4.x	structure-3.1.1.jira6.jar[1018]	2016-06-02	2017-06-02	8aaaefbbec05f1d4a81d6ddc382a6ffa
3.1.0.jira7	Jira 7.0+	structure-3.1.0.jira7.jar[1019]	2016-04-30	2017-04-30	5110b83c9ea8601a0e2a64eee292769a

1009 https://d1.almworks.com/.files/structure-3.3.0.jira7.jar
1010 https://d1.almworks.com/.files/structure-3.3.0.jira6.jar
1011 https://d1.almworks.com/.files/structure-3.2.3.jira6.jar
1012 https://d1.almworks.com/.files/structure-3.2.2.jira6.jar
1013 https://d1.almworks.com/.files/structure-3.2.1.jira7.jar
1014 https://d1.almworks.com/.files/structure-3.2.1.jira6.jar
1015 https://d1.almworks.com/.files/structure-3.2.0.jira7.jar
1016 https://d1.almworks.com/.files/structure-3.2.0.jira6.jar
1017 https://d1.almworks.com/.files/structure-3.1.1.jira7.jar
1018 https://d1.almworks.com/.files/structure-3.1.1.jira6.jar
1019 https://d1.almworks.com/.files/structure-3.1.0.jira7.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
3.1.0.jira6	Jira 6.3 – 6.4.x	structure-3.1.0.jira6.jar[1020]	2016-04-30	2017-04-30	f87b93ea163ae33f6a920a3171ed148a
3.0.2.jira7	Jira 7.0+	structure-3.0.2.jira7.jar[1021]	2016-04-26	2017-04-26	90b8f5f8f900a3498baa09493cdbc1bd
3.0.2.jira6	Jira 6.3 – 6.4.x	structure-3.0.2.jira6.jar[1022]	2016-04-26	2017-04-26	1bbb2615d76091022f00e07be5d03693
3.0.1.jira7	Jira 7.0+	structure-3.0.1.jira7.jar[1023]	2016-04-11	2017-04-11	929fcfdac6c4e939596499f2d11535cc
3.0.1.jira6	Jira 6.3 – 6.4.x	structure-3.0.1.jira6.jar[1024]	2016-04-11	2017-04-11	9ffc3f78d653f22564b6badd6c226ffb
3.0.0.jira7	Jira 7.0+	structure-3.0.0.jira7.jar[1025]	2016-04-05	2017-04-05	3937e3c80511bcbf4cd11d551335d2fc
3.0.0.jira6	Jira 6.3 – 6.4.x	structure-3.0.0.jira6.jar[1026]	2016-04-05	2017-04-05	ce7590fa0af69b9d031345fbb0869a15
2.11.2.jira7	Jira 7.0+	structure-2.11.2.jira7.jar[1027]	2016-02-25	2017-02-25	9b182941865f816229d7bcb7a4a0f251
2.11.2.jira6	Jira 6.3 – 6.4.x	structure-2.11.2.jira6.jar[1028]	2016-02-25	2017-02-25	44d8c2c693b4f8e10aeec2bc8fd9392f
2.11.1.jira7	JIRA 7.0+	structure-2.11.1.jira7.jar[1029]	2015-12-16	2016-12-16	684e320a7d7ad6b2974cc403e34fd84b
2.11.1.jira6	JIRA 6.3 – 6.4.x	structure-2.11.1.jira6.jar[1030]	2015-12-16	2016-12-16	7846f40430c1170b0d00c72d3758bb5d
2.11.0.jira7	JIRA 7.0+	structure-2.11.0.jira7.jar[1031]	2015-10-16	2016-10-16	bbe3492315892c5d603396ea2a0f4f58

1020 https://d1.almworks.com/.files/structure-3.1.0.jira6.jar
1021 https://d1.almworks.com/.files/structure-3.0.2.jira7.jar
1022 https://d1.almworks.com/.files/structure-3.0.2.jira6.jar
1023 https://d1.almworks.com/.files/structure-3.0.1.jira7.jar
1024 https://d1.almworks.com/.files/structure-3.0.1.jira6.jar
1025 https://d1.almworks.com/.files/structure-3.0.0.jira7.jar
1026 https://d1.almworks.com/.files/structure-3.0.0.jira6.jar
1027 https://d1.almworks.com/.files/structure-2.11.2.jira7.jar
1028 https://d1.almworks.com/.files/structure-2.11.2.jira6.jar
1029 https://d1.almworks.com/.files/structure-2.11.1.jira7.jar
1030 https://d1.almworks.com/.files/structure-2.11.1.jira6.jar
1031 https://d1.almworks.com/.files/structure-2.11.0.jira7.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
2.11.0.jira6	JIRA 6.3 – 6.4.x	structure-2.11.0.jira6.jar[1032]	2015-10-16	2016-10-16	6a6ac210e7ed862be62cc1e71e5adbdd
2.10.6	JIRA 6.1 – 6.4.x	structure-2.10.6.jar[1033]	2015-07-03	2016-07-03	61947f5b97217d31d2926104231644e1
2.10.5	JIRA 6.1 – 6.4.x	structure-2.10.5.jar[1034]	2015-06-04	2016-06-04	44849ddcc940dd34b6cf862df91d43cd
2.10.4	JIRA 6.1 – 6.4.x	structure-2.10.4.jar[1035]	2015-05-29	2016-05-29	bb435e06f769df66981000751b9b09e4
2.10.3	JIRA 6.1 – 6.4.x	structure-2.10.3.jar[1036]	2015-04-30	2016-04-30	1b4e3ce3b2c335a4fe05b557f4af9e65
2.10.2	JIRA 6.1 – 6.4.x	structure-2.10.2.jar[1037]	2015-04-27	2016-04-27	f03c9ee599c28d21a96a0d400d7dcf9c
2.10.1	JIRA 6.1 – 6.4.x	structure-2.10.1.jar[1038]	2015-03-26	2016-03-26	5f489fe1dcc491075f8c496e31e8c283
2.10.0	JIRA 6.1 – 6.3.15	structure-2.10.0.jar[1039]	2015-02-23	2016-02-23	c943296433b01fe8ab74d25fffc44ff5
2.9.1	JIRA 6.1 – 6.3+	structure-2.9.1.jar[1040]	2014-10-28	2015-10-28	1fb9cdf5d612c8ce9711293b98c299ea
2.9.0	JIRA 6.1 – 6.3+	structure-2.9.0.jar[1041]	2014-09-08	2015-09-08	b03dfb5081faf76c9fd4fe043e2d12bd
2.8.3	JIRA 6.1 – 6.3+	structure-2.8.3.jar[1042]	2014-07-25	2015-07-25	c54909cc9bc02e4c41caf76f622ff90e

1032 https://d1.almworks.com/.files/structure-2.11.0.jira6.jar
1033 https://d1.almworks.com/.files/structure-2.10.6.jar
1034 https://d1.almworks.com/.files/structure-2.10.5.jar
1035 https://d1.almworks.com/.files/structure-2.10.4.jar
1036 https://d1.almworks.com/.files/structure-2.10.3.jar
1037 https://d1.almworks.com/.files/structure-2.10.2.jar
1038 https://d1.almworks.com/.files/structure-2.10.1.jar
1039 https://d1.almworks.com/.files/structure-2.10.0.jar
1040 https://d1.almworks.com/.files/structure-2.9.1.jar
1041 https://d1.almworks.com/.files/structure-2.9.0.jar
1042 https://d1.almworks.com/.files/structure-2.8.3.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
2.8.2	JIRA 6.1 – 6.3+	structure-2.8.2.jar[1043]	2014-07-21	2015-07-21	e14ecebefb954b8275c3096fab7568af
2.8.1	JIRA 6.1 – 6.2.7	structure-2.8.1.jar[1044]	2014-07-17	2015-07-17	40a021f938154a5bd38e4f66151c9a40
2.8.0	JIRA 6.1 – 6.3+	structure-2.8.0.jar[1045]	2014-07-07	2015-07-07	ba56673401c7a818297b153a23c2ca19
2.7.0	JIRA 6.0 – 6.2+	structure-2.7.0.jar[1046]	2014-02-26	2015-02-26	7d59559069ee957848452e9dee569ab7
2.6.1	JIRA 5.2 – 5.2.11	structure-2.6.1.jira5.jar[1047]	2014-02-25	2015-02-25	595b6a723eb3e1cd3a70f38ff7bef81b
2.6.0	JIRA 6.0 – 6.1+	structure-2.6.0.jar[1048]	2014-01-24	2015-01-24	dc5a9d344d7ace5176f7f1cde6bd6e9c
2.6.0	JIRA 5.2 – 5.2.11	structure-2.6.0.jira5.jar[1049]	2014-01-24	2015-01-24	ee4a0b0c8e91c122503b3dc7554b9815
2.5.3	JIRA 6.0 – 6.1+	structure-2.5.3.jar[1050]	2013-11-22	2014-11-22	df35dc5440d6287402bd0afabf7afb9f
2.5.3	JIRA 5.2 – 5.2.11	structure-2.5.3.jira5.jar[1051]	2013-11-22	2014-11-22	24232bf5572ceeb0b9b9af7a56214340

1043 https://d1.almworks.com/.files/structure-2.8.2.jar
1044 https://d1.almworks.com/.files/structure-2.8.1.jar
1045 https://d1.almworks.com/.files/structure-2.8.0.jar
1046 https://d1.almworks.com/.files/structure-2.7.0.jar
1047 https://d1.almworks.com/.files/structure-2.6.1.jira5.jar
1048 https://d1.almworks.com/.files/structure-2.6.0.jar
1049 https://d1.almworks.com/.files/structure-2.6.0.jira5.jar
1050 https://d1.almworks.com/.files/structure-2.5.3.jar
1051 https://d1.almworks.com/.files/structure-2.5.3.jira5.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
2.5.2	JIRA 6.0 – 6.1+	structure-2.5.2.jar[1052]	2013-11-01	2014-11-01	76e72d8384df35c2e3f4e077f4621d8e
2.5.2	JIRA 5.2 – 5.2.11	structure-2.5.2.jira5.jar[1053]	2013-11-01	2014-11-01	6481fee6eeeb3a817fb510a582983540
2.5.1	JIRA 6.0 – 6.1+	structure-2.5.1.jar[1054]	2013-10-30	2014-10-30	6b460040bf22eced9f03f0f4fe7133d7
2.5.1	JIRA 5.2 – 5.2.11	structure-2.5.1.jira5.jar[1055]	2013-10-30	2014-10-30	7322b4d016451a8a2883efb79d545219
2.5.0	JIRA 6.0 – 6.1+	structure-2.5.0.jar[1056]	2013-09-26	2014-09-26	a0c6110c7696f8f95241480e5acfe509
2.5.0	JIRA 5.2 – 5.2.11	structure-2.5.0.jira5.jar[1057]	2013-09-26	2014-09-26	e6d010d9f8d77127c58ac086be30746a
2.4.3	JIRA 6.0+	structure-2.4.3.jar[1058]	2013-09-01	2014-09-01	aa8351adff97070088503e8fbb89b9dd
2.4.3	JIRA 5.0.1 – 5.2+	structure-2.4.3.jira5.jar[1059]	2013-09-01	2014-09-01	f6a9c660777e92a67cc3ad704e4f9304
2.4.2	JIRA 6.0+	structure-2.4.2.jar[1060]	2013-08-07	2014-08-07	548b80001bcdb3c9df1ba9c0ddaa687d

1052 https://d1.almworks.com/.files/structure-2.5.2.jar
1053 https://d1.almworks.com/.files/structure-2.5.2.jira5.jar
1054 https://d1.almworks.com/.files/structure-2.5.1.jar
1055 https://d1.almworks.com/.files/structure-2.5.1.jira5.jar
1056 https://d1.almworks.com/.files/structure-2.5.0.jar
1057 https://d1.almworks.com/.files/structure-2.5.0.jira5.jar
1058 https://d1.almworks.com/.files/structure-2.4.3.jar
1059 https://d1.almworks.com/.files/structure-2.4.3.jira5.jar
1060 https://d1.almworks.com/.files/structure-2.4.2.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
2.4.2	JIRA 5.0.1 – 5.2+	structure-2.4.2.jira5.jar[1061]	2013-08-07	2014-08-07	aaa4a979b3da658806b87b5d260e4517
2.4.1	JIRA 6.0+	structure-2.4.1.jar[1062]	2013-08-02	2014-08-02	8f7e92dad13a40a8bc630b26dae956da
2.4.1	JIRA 5.0.1 – 5.2+	structure-2.4.1.jira5.jar[1063]	2013-08-02	2014-08-02	ee875d01f47c57bea1f2770607b3edec
2.4.0	JIRA 6.0+	structure-2.4.0.jar[1064]	2013-06-14	2014-06-14	66fa84503c5e787d4643a8809754daac
2.4.0	JIRA 5.0.1 – 5.2+	structure-2.4.0.jira5.jar[1065]	2013-06-14	2014-06-14	ad5f88295185af4642083efc10a6b27d
2.3.0	JIRA 6.0+	structure-2.3.0.jar[1066]	2013-05-18	2014-05-18	329921b60229a0f623b36001e7c4f67c
2.3.0	JIRA 5.0 – 5.2+	structure-2.3.0.jira5.jar[1067]	2013-05-18	2014-05-18	deae862ca01f0ff02103387a4796d371
2.2.1	JIRA 5.0 – 5.2+	structure-2.2.1.jar[1068]	2013-04-12	2014-04-12	83b6f08e131c95c5298b188bd08e01a4
2.2.0	JIRA 5.0 – 5.2+	structure-2.2.0.jar[1069]	2013-03-28	2014-03-28	66e9d6340f31a7c245866ef8849ea8ea

1061 https://d1.almworks.com/.files/structure-2.4.2.jira5.jar
1062 https://d1.almworks.com/.files/structure-2.4.1.jar
1063 https://d1.almworks.com/.files/structure-2.4.1.jira5.jar
1064 https://d1.almworks.com/.files/structure-2.4.0.jar
1065 https://d1.almworks.com/.files/structure-2.4.0.jira5.jar
1066 https://d1.almworks.com/.files/structure-2.3.0.jar
1067 https://d1.almworks.com/.files/structure-2.3.0.jira5.jar
1068 https://d1.almworks.com/.files/structure-2.2.1.jar
1069 https://d1.almworks.com/.files/structure-2.2.0.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
2.1.1	JIRA 5.0 – 5.2+	structure-2.1.1.jar[1070]	2013-03-18	2014-03-18	`309396fef173778035bfe60d1d210c61`
2.1.0	JIRA 5.0 – 5.2+	structure-2.1.0.jar[1071]	2013-02-15	2014-02-15	`fff88a32d389171280a39c5792636089`
2.0.0	JIRA 5.0 – 5.2+	structure-2.0.0.jar[1072]	2012-11-19	2013-11-19	`fb08820d0bad6797219343f9ce6e40e4`
2.0.0	JIRA 4.4	structure-2.0.0.jira44.jar[1073]	2012-11-19	2013-11-19	`54343d964ccbf2107ab43d0e27a1da20`
1.7.1	JIRA 5.0 – 5.1	structure-1.7.1.jar[1074]	2012-08-07	2013-08-07	`cc4dd0808905ed27750e9b1d23ba8524`
1.7.0	JIRA 5.0 – 5.1	structure-1.7.0.jar[1075]	2012-07-11	2013-07-11	`2ef3a16f31ad6974c53a5104002dab0b`
1.7.0	JIRA 4.4	structure-1.7.0.jira44.jar[1076]	2012-07-11	2013-07-11	`1abbaf16cd918a9a77df54210151be14`
1.6.0	JIRA 5.0	structure-1.6.0.jar[1077]	2012-05-12	2013-05-12	`72faf0db1e10ddf9da4c34be0e8abc57`
1.6.0	JIRA 4.4	structure-1.6.0.jira44.jar[1078]	2012-05-12	2013-05-12	`0d7ce053e6114c73f36fee4419b7e8c0`

1070 https://d1.almworks.com/.files/structure-2.1.1.jar
1071 https://d1.almworks.com/.files/structure-2.1.0.jar
1072 https://d1.almworks.com/.files/structure-2.0.0.jar
1073 https://d1.almworks.com/.files/structure-2.0.0.jira44.jar
1074 https://d1.almworks.com/.files/structure-1.7.1.jar
1075 https://d1.almworks.com/.files/structure-1.7.0.jar
1076 https://d1.almworks.com/.files/structure-1.7.0.jira44.jar
1077 https://d1.almworks.com/.files/structure-1.6.0.jar
1078 https://d1.almworks.com/.files/structure-1.6.0.jira44.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
1.6.0	JIRA 4.3	structure-1.6.0.jira43.jar[1079]	2012-05-12	2013-05-12	73cd71589c0852cf175a1f1ca5f7ba54
1.5.2	JIRA 5.0	structure-1.5.2.jar[1080]	2012-03-16	2013-03-16	9a456039c086212b16a946411fea1f2e
1.5.2	JIRA 4.4	structure-1.5.2.jira44.jar[1081]	2012-03-16	2013-03-16	23123dbcbdb4eff3f42668622d34ad59
1.5.2	JIRA 4.3	structure-1.5.2.jira43.jar[1082]	2012-03-16	2013-03-16	e1fbeee8bde69e580e53fda801126fb5
1.5.1	JIRA 5.0	structure-1.5.1.jar[1083]	2012-02-22	2013-02-22	d6552ca8da9366268b30bbf2b4505747
1.5.0	JIRA 5.0	structure-1.5.0.jar[1084]	2012-02-17	2013-02-17	e82766909cb1eeed255e051045c56166
1.5.0	JIRA 4.4	structure-1.5.0.jira44.jar[1085]	2012-02-17	2013-02-17	4f753e1700836d2f1eb61e99bd756bcb
1.5.0	JIRA 4.3	structure-1.5.0.jira43.jar[1086]	2012-02-17	2013-02-17	9335abcec6f435385a2f82a029148bc6
1.4.1	JIRA 4.4	structure-1.4.1.jar[1087]	2012-01-16	2013-01-16	838276cf9ae777a5f4e072df56fad7fb

1079 https://d1.almworks.com/.files/structure-1.6.0.jira43.jar
1080 https://d1.almworks.com/.files/structure-1.5.2.jar
1081 https://d1.almworks.com/.files/structure-1.5.2.jira44.jar
1082 https://d1.almworks.com/.files/structure-1.5.2.jira43.jar
1083 https://d1.almworks.com/.files/structure-1.5.1.jar
1084 https://d1.almworks.com/.files/structure-1.5.0.jar
1085 https://d1.almworks.com/.files/structure-1.5.0.jira44.jar
1086 https://d1.almworks.com/.files/structure-1.5.0.jira43.jar
1087 https://d1.almworks.com/.files/structure-1.4.1.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
1.4.1	JIRA 4.3	structure-1.4.1.jira43.jar[1088]	2012-01-16	2013-01-16	`c15bd779e874704d624cff68eed54bf5`
1.4.1	JIRA 4.2	structure-1.4.1.jira42.jar[1089]	2012-01-16	2013-01-16	`c4b8cd057128dd66ba293b738947d57e`
1.4	JIRA 4.4	structure-1.4.0.jar[1090]	2012-01-08	2013-01-08	`cbf9a5b32305b252ad591e7a0722cb1f`
1.4	JIRA 4.3	structure-1.4.0.jira43.jar[1091]	2012-01-08	2013-01-08	`01d0bf95e1f40a70365b2b260d92d360`
1.4	JIRA 4.2	structure-1.4.0.jira42.jar[1092]	2012-01-08	2013-01-08	`1bfbda79f93e24ea55257c29ec7a1f60`
1.3.2	JIRA 4.4	structure-1.3.2.jar[1093]	2011-11-29	2012-11-29	`091bbea96a1f757e86b47ab6302ee506`
1.3.2	JIRA 4.3	structure-1.3.2.jira43.jar[1094]	2011-11-29	2012-11-29	`38a7481dc204046e6c52956e79a4ab14`
1.3.2	JIRA 4.2	structure-1.3.2.jira42.jar[1095]	2011-11-29	2012-11-29	`bafb5476b1f5c73486d48a21e961d175`
1.3.1	JIRA 4.4	structure-1.3.1.jar[1096]	2011-11-10	2012-11-10	`d426687615b34014ddace26aded9588a`

1088 https://d1.almworks.com/.files/structure-1.4.1.jira43.jar
1089 https://d1.almworks.com/.files/structure-1.4.1.jira42.jar
1090 https://d1.almworks.com/.files/structure-1.4.0.jar
1091 https://d1.almworks.com/.files/structure-1.4.0.jira43.jar
1092 https://d1.almworks.com/.files/structure-1.4.0.jira42.jar
1093 https://d1.almworks.com/.files/structure-1.3.2.jar
1094 https://d1.almworks.com/.files/structure-1.3.2.jira43.jar
1095 https://d1.almworks.com/.files/structure-1.3.2.jira42.jar
1096 https://d1.almworks.com/.files/structure-1.3.1.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
1.3.1	JIRA 4.3	structure-1.3.1.jira43.jar[1097]	2011-11-10	2012-11-10	a778f7f72f6a76d364064121fc3c02fb
1.3.1	JIRA 4.2	structure-1.3.1.jira42.jar[1098]	2011-11-10	2012-11-10	51048202d1671cf4b4a0ffeadd9d89a9
1.3	JIRA 4.4	structure-1.3.0.jar[1099]	2011-10-24	2012-10-24	c0cc1db5ba33648ee29b66e5f3585bab
1.3	JIRA 4.3	structure-1.3.0.jira43.jar[1100]	2011-10-24	2012-10-24	79ecd34177d55ce4c01456ec5216ac3f
1.3	JIRA 4.2	structure-1.3.0.jira42.jar[1101]	2011-10-24	2012-10-24	13430ccee213530ee8bf0e98c24f80dc
1.2	JIRA 4.4	structure-1.2.0.jar[1102]	2011-08-30	2012-08-30	9be530ae1f9e68b61afdc936ab4bfc50
1.2	JIRA 4.3	structure_jira4.3backport-1.2.0.jar[1103]	2011-08-30	2012-08-30	9a3a31a0c202f80c9d05f07cb5a08882
1.2	JIRA 4.2	structure_jira4.2backport-1.2.0.jar[1104]	2011-08-30	2012-08-30	7f9bc6f83028adcb83c2eb436723c33a
1.1.1	JIRA 4.4	structure-1.1.1.jar[1105]	2011-08-08	2012-08-08	6d0a9245a4c1c6de92eb076295ba7393

1097 https://d1.almworks.com/.files/structure-1.3.1.jira43.jar
1098 https://d1.almworks.com/.files/structure-1.3.1.jira42.jar
1099 https://d1.almworks.com/.files/structure-1.3.0.jar
1100 https://d1.almworks.com/.files/structure-1.3.0.jira43.jar
1101 https://d1.almworks.com/.files/structure-1.3.0.jira42.jar
1102 https://d1.almworks.com/.files/structure-1.2.0.jar
1103 https://d1.almworks.com/.files/structure_jira4.3backport-1.2.0.jar
1104 https://d1.almworks.com/.files/structure_jira4.2backport-1.2.0.jar
1105 https://d1.almworks.com/.files/structure-1.1.1.jar

Structure Version	Jira version	Download link	Publish Date	End of Life	MD5
1.1.1	JIRA 4.3	structure_jira4.3backport-1.1.1.jar[1106]	2011-08-08	2012-08-08	22252d97bef859fb7f442e0c9199089f
1.1.1	JIRA 4.2	structure_jira4.2backport-1.1.1.jar[1107]	2011-08-08	2012-08-08	f6dc10634dd0a4d02029018ddf3c7df6
1.1	JIRA 4.3	structure-1.1.jar[1108]	2011-07-29	2012-07-29	7c5d17b69e6d90320693648651a1c753
1.1	JIRA 4.2	structure_jira4.2backport-1.1.jar[1109]	2011-07-29	2012-07-29	959a9779540428f9325464a4e45714cc
1.1	JIRA 4.1	structure_jira4.1backport-1.1.jar[1110]	2011-07-29	2012-07-29	b06ac14cafb69df6f6394b8e07d5ec56
1.1	JIRA 4.4 Preview	structure_jira4.4preview-1.1.jar[1111]	2011-07-29	2011-10-29	c6796a4f3b2392c61b660d03311ebde1
1.0	JIRA 4.3	structure-1.0.jar[1112]	2011-07-13	2012-07-13	
1.0	JIRA 4.2	structure_jira4.2backport-1.0.jar[1113]	2011-07-13	2012-07-13	
1.0	JIRA 4.1	structure_jira4.1backport-1.0.jar[1114]	2011-07-13	2012-07-13	

1106 https://d1.almworks.com/.files/structure_jira4.3backport-1.1.1.jar
1107 https://d1.almworks.com/.files/structure_jira4.2backport-1.1.1.jar
1108 https://d1.almworks.com/.files/structure-1.1.jar
1109 https://d1.almworks.com/.files/structure_jira4.2backport-1.1.jar
1110 https://d1.almworks.com/.files/structure_jira4.1backport-1.1.jar
1111 https://d1.almworks.com/.files/structure_jira4.4preview-1.1.jar
1112 https://d1.almworks.com/.files/structure-1.0.jar
1113 https://d1.almworks.com/.files/structure_jira4.2backport-1.0.jar
1114 https://d1.almworks.com/.files/structure_jira4.1backport-1.0.jar

End of Life:

- For beta versions: Beta versions are scheduled to expire after certain amount of time. You will need to upgrade to another Beta or Production version before that date.
- For other versions: You will be able to use this version of the plugin indefinitely, however, after End-of-Life date, the support for the version is limited.