

---

# Structure for Jira Documentation

---

**com.** Igor Sereda

**k15t.**

**scroll.**

**pdf.**

**template.**

**default.**

**titlepage.**

**author**

**com.** 1

**k15t.**

**scroll.**

**pdf.**

**template.**

**default.**

**titlepage.**

**version**

**com.** Jun 3, 2020 3:52 PM

# k15t. Table of Contents

scroll.

pdf.

template.

default.

titlepage.

date

<b>1</b>	<b>Structure Quick Start Guide</b>	
1.1	Before You Begin	
1.2	Creating Your First Structure	
1.2.1	Your First Structure	
1.2.2	Next Steps	
1.2.3	Building a Structure with Automation	
1.2.4	Building a Structure Manually	24
<b>1.3</b>	<b>Working with Structure</b>	<b>27</b>
1.3.1	Moving, Adding and Removing Items from a Structure	27
1.3.2	Adding Columns	29
1.3.3	The Structure Board	30
1.3.4	Structure in Other Locations	30
1.3.5	Next Step	31
1.3.6	Working with Issues in Structure	31
1.3.7	Search Filter and Sort	33
<b>1.4</b>	<b>Help and Support</b>	<b>36</b>
1.4.1	Available Resources	36
1.4.2	Resources Within Structure	36
<b>1.5</b>	<b>Structure.Pages Quick Start Guide</b>	<b>37</b>
1.5.1	Basic Facts	37
1.5.2	Installation	37
1.5.3	Main Functionality	39
<b>2</b>	<b>Structure User's Guide</b>	<b>42</b>
<b>2.1</b>	<b>Navigating Structure</b>	<b>42</b>
2.1.1	Basic Concepts	42
2.1.2	Structure Menu	43
2.1.3	Main Structure Toolbar	45
2.1.4	Structure Widget Overview	47
2.1.5	Jira Pages with Structure	54
2.1.6	Sharing a Perspective	65
<b>2.2</b>	<b>Basic Operations</b>	<b>67</b>
2.2.1	Adding Items to a Structure	67
2.2.2	Moving Items within Structure	73
2.2.3	Removing Items from Structure	75
2.2.4	Undoing Changes	76
2.2.5	Working with Issues	76
<b>2.3</b>	<b>Automation</b>	<b>93</b>
2.3.1	How Automation Works	94
2.3.2	Types of Generators	94
2.3.3	Generator Scope	94
2.3.4	Adding a Generator	94

2.3.5	Types of Generators	95
2.3.6	Effectors	128
2.3.7	Generator Scope	139
2.3.8	Editing a Generator	141
2.3.9	Deleting a Generator	142
2.3.10	Paused Generation	142
2.3.11	Manual Adjustments	146
2.3.12	Order of Operation for Generators	151
<b>2.4</b>	<b>Search, Filter and Transformation</b>	<b>152</b>
2.4.1	Search	152
2.4.2	Filter	155
2.4.3	Transformations	156
2.4.4	Pinned Item Mode	166
2.4.5	Identifying Duplicate Items	168
<b>2.5</b>	<b>Formulas</b>	<b>171</b>
2.5.1	Formula Columns	171
2.5.2	Wiki Markup in Formula Columns	182
2.5.3	Bundled Formulas	193
2.5.4	Expr Language	195
<b>2.6</b>	<b>Structured JQL</b>	<b>249</b>
2.6.1	Get Started with S-JQL	249
2.6.2	Master S-JQL	249
2.6.3	S-JQL Cookbook	249
2.6.4	S-JQL Reference	255
2.6.5	structure() JQL function	279
<b>2.7</b>	<b>Columns and Views</b>	<b>281</b>
2.7.1	Structure Columns	282
2.7.2	Configuring View	319
2.7.3	Managing Views	329
2.7.4	Displaying Full Cell Content	337
2.7.5	Two-Panel Mode	338
2.7.6	Full Screen Mode	341
<b>2.8</b>	<b>Managing Structures</b>	<b>342</b>
2.8.1	Locating a Structure	344
2.8.2	Default Structure	345
2.8.3	Favorite Structures	346
2.8.4	Structure Details	347
2.8.5	Customizing View Settings	349

2.8.6	Structure Permissions	351
2.8.7	Creating New Structures	355
2.8.8	Copying a Structure	355
2.8.9	Archiving a Structure	363
2.8.10	Deleting a Structure	366
2.8.11	Template Structures and Projects	367
2.8.12	Viewing History of a Structure	368
2.8.13	Printing Structure	370
2.8.14	Exporting Structure to XLS (Excel)	371
2.8.15	Real-Time Collaboration	374
2.8.16	Structure Activity Stream	374
<b>2.9</b>	<b>Keyboard Shortcuts</b>	<b>378</b>
2.9.1	Keyboard Shortcuts (PC)	378
2.9.2	Keyboard Shortcuts (Mac)	381
2.9.3	Quick Action Lookup	383
<b>2.10</b>	<b>Structure Gadget</b>	<b>383</b>
2.10.1	Dashboard Gadget	383
2.10.2	Confluence Gadget	388
<b>2.11</b>	<b>Getting Help</b>	<b>395</b>
2.11.1	Resources Within Structure	395
2.11.2	Contact Us	396
<b>3</b>	<b>Structure Administrator's Guide</b>	<b>397</b>
<b>3.1</b>	<b>Installing Structure</b>	<b>397</b>
3.1.1	Migrating Data from Structure 2 to Structure 3	398
3.1.2	Memory Guidelines	399
3.1.3	Uninstalling and Reinstalling Structure	402
3.1.4	Upgrading and Downgrading	402
<b>3.2</b>	<b>Setting Up Structure License</b>	<b>405</b>
3.2.1	Setting Up Evaluation License	405
3.2.2	Licenses from ALM Works and from Atlassian	405
3.2.3	Purchasing a Commercial License	407
3.2.4	Migrating Licenses	407
3.2.5	Structure License Parameters	408
3.2.6	When Structure is Available for Free	408
3.2.7	License Maintenance and Expiration	409

<b>3.3</b>	<b>Selecting Structure-Enabled Projects</b>	<b>409</b>
<b>3.4</b>	<b>Global Permissions</b>	<b>410</b>
3.4.1	Who Has Access to the Structure	410
3.4.2	Restricting User Access to Structure	410
3.4.3	Changing Permission to Create New Structures	412
3.4.4	Changing Permission to Access Automation	413
3.4.5	Changing Permissions to Configure and Run Effectors	413
<b>3.5</b>	<b>Changing Structure Defaults</b>	<b>414</b>
3.5.1	Initial Configuration	414
3.5.2	Changing Default Structure	415
3.5.3	Changing Default View Settings	416
3.5.4	Changing Default Options for the Issue and Project Pages	416
<b>3.6</b>	<b>Attribute Sensitivity Settings</b>	<b>417</b>
3.6.1	How Does Attribute Sensitivity Work?	418
3.6.2	Setting the Attribute Sensitivity Mode	418
3.6.3	Assigning Specific Non-Sensitive Attributes	419
<b>3.7</b>	<b>Structure Backup, Restore and Migration</b>	<b>420</b>
3.7.1	Using Structure Backup	420
3.7.2	Backing Up Structure	421
3.7.3	Restoring Structure from Backup	421
3.7.4	Migrating Structures	423
<b>3.8</b>	<b>Automatic Structure Maintenance</b>	<b>425</b>
3.8.1	Automatic Structure Maintenance	425
3.8.2	Maintenance Schedule	425
3.8.3	Maintenance Tasks	427
3.8.4	Running Maintenance Tasks Manually	429
<b>3.9</b>	<b>Workflow Integration</b>	<b>429</b>
3.9.1	Structure Workflow Validator	429
3.9.2	Structure Workflow Condition	431
<b>3.10</b>	<b>Running Structure on Jira Data Center</b>	<b>431</b>
3.10.1	Archived Projects and Structure	432
<b>3.11</b>	<b>Anonymous Usage Statistics</b>	<b>432</b>
3.11.1	Viewing Current Statistics	433
3.11.2	Turning Anonymous Usage Statistics On and Off	433
3.11.3	Structure Usage Statistics	433
<b>3.12</b>	<b>Structure Files</b>	<b>434</b>

3.12.1	\$JIRA_HOME/structure	434
<b>3.13</b>	<b>Turning Off Optional Features</b>	<b>434</b>
<b>3.14</b>	<b>Advanced Configuration and Dark Features</b>	<b>435</b>
3.14.1	Setting Application Properties with the Structure Dark Features and Fine Tuning Interface	435
3.14.2	Setting System Properties	436
3.14.3	Structure size limit	437
3.14.4	Structure Automation limits	438
3.14.5	Automation Defaults	438
3.14.6	Manual adjustments	438
3.14.7	Hidden Issue Links	439
3.14.8	Index Consistency Checks	439
3.14.9	Alternative initial values for project/type when creating an issue in dialog	440
3.14.10	Structured Predictions	444
3.14.11	Synchronization	451
3.14.12	Structure 6.0 EAP	486
<b>3.15</b>	<b>System Requirements</b>	<b>486</b>
3.15.1	Atlassian Platform	486
3.15.2	Databases	486
3.15.3	Browsers	486
3.15.4	Server Requirements	487
3.15.5	Non-Conforming systems	487
<b>3.16</b>	<b>Confluence Gadget - Admin Configuration</b>	<b>487</b>
3.16.1	Adding Structure Gadget to Confluence Configuration	488
3.16.2	Setting Up CORS Filter in JIRA	489
<b>3.17</b>	<b>Best Practices</b>	<b>490</b>
3.17.1	Best Practices and Performance Considerations	490
3.17.2	Backup Strategy	508
3.17.3	Gradual Deployment	509
<b>3.18</b>	<b>ScriptRunner and Structure Cookbook</b>	<b>510</b>
3.18.1	Use-cases	510
3.18.2	Automatically Remove Issues Based on JQL Query	511
3.18.3	Bulk Change Owners of Structures and Generators	514
3.18.4	Creating a New Structure Programmatically	520
3.18.5	Creating Generators	521
3.18.6	Show All Structure Boards and Corresponding Item Counts	535
3.18.7	Show Related Issues in a Separate Column	536

3.18.8	Show work logged per user and issue for a structure	538
3.18.9	Updating a field (ex. label) when checking all issues against a JQL query	544
<b>3.19</b>	<b>Clearing Structure Caches</b>	<b>545</b>
<b>3.20</b>	<b>Anonymizing Users</b>	<b>546</b>
3.20.1	Review User's Structures	546
3.20.2	Transfer Ownership	547
<b>4</b>	<b>Structure Developer's Guide</b>	<b>548</b>
<b>4.1</b>	<b>Structure Developer Documentation</b>	<b>548</b>
<b>4.2</b>	<b>Structure Concepts, Developer's Perspective</b>	<b>548</b>
4.2.1	Basic Concepts Overview	549
4.2.2	A Note on Extensibility	550
<b>4.3</b>	<b>Accessing Structure from JIRA Plugin</b>	<b>550</b>
4.3.1	Setting Up the Integration	550
4.3.2	Structure Services	555
4.3.3	Building Forest Specification	557
4.3.4	Reading Structure Content	558
4.3.5	Changing Structure Content	561
4.3.6	Loading Attribute Values	565
4.3.7	Creating and Adding Folders	567
4.3.8	Creating Dynamic Structures	568
<b>4.4</b>	<b>Extending Structure Functionality</b>	<b>569</b>
4.4.1	Creating a New Column Type	569
4.4.2	Loading Additional Web Resources For Structure Widget	590
4.4.3	Declaring a New Generic Item Type	591
<b>4.5</b>	<b>Accessing Structure Data Remotely</b>	<b>594</b>
<b>4.6</b>	<b>Reference</b>	<b>594</b>
4.6.1	Structure Developer Reference	594
4.6.2	Structure Java API Reference	594
4.6.3	Structure Plugin Module Types	597
4.6.4	Structure REST API Reference	606
4.6.5	Structure JavaScript API Reference	646
4.6.6	Web Resource Contexts	668
<b>4.7</b>	<b>API Usage Samples</b>	<b>668</b>
4.7.1	Download	668

4.7.2	Example List	669
<b>4.8</b>	<b>Archived Documentation</b>	<b>670</b>
4.8.1	Structure 3 API Changes	670
4.8.2	Creating a New Synchronizer	678
4.8.3	structure-synchronizer	679
<b>5</b>	<b>Structure FAQ</b>	<b>681</b>
<b>5.1</b>	<b>Frequently Asked Questions</b>	<b>681</b>
<b>5.2</b>	<b>Data Center Approved Apps FAQ</b>	<b>681</b>
5.2.1	What are Data Center approved apps?	681
5.2.2	Didn't Structure already work with Data Center?	681
5.2.3	Why has Structure made this change?	682
5.2.4	What are the criteria for being Data Center approved?	682
5.2.5	Why should I upgrade to the Data Center approved app version?	682
5.2.6	Why is the price different for Data Center approved apps?	683
5.2.7	Do I have to switch to the Data Center approved version? Can I continue to use the server app in Data Center?	683
<b>5.3</b>	<b>Cannot Create an Issue With +Next Issue (+Sub-Issue) Because of the Required Fields</b>	<b>683</b>
5.3.1	Question	683
5.3.2	Answer	683
<b>5.4</b>	<b>Add-on Manager Says Structure Is Unlicensed</b>	<b>684</b>
5.4.1	Question	684
5.4.2	Answer	684
<b>5.5</b>	<b>No Check Mark Displayed for a Resolved Issue</b>	<b>684</b>
5.5.1	Question	684
5.5.2	Answer	685
<b>5.6</b>	<b>Structure Won't Start</b>	<b>686</b>
5.6.1	Question	686
5.6.2	Answer	686
<b>5.7</b>	<b>After an Issue is Moved to Another Project, It Cannot Be Found in the Structure</b>	<b>689</b>
5.7.1	Question	689
5.7.2	Answer	689
<b>5.8</b>	<b>User Cannot Access Structure</b>	<b>689</b>

5.8.1	Question	689
5.8.2	Answer	689
<b>5.9</b>	<b>Where Do I Find the Jira Server ID?</b>	<b>690</b>
<b>5.10</b>	<b>Using Subtasks and Structure</b>	<b>690</b>
5.10.1	Question	690
5.10.2	Answer	690
<b>5.11</b>	<b>Difference from Sub-tasks</b>	<b>691</b>
5.11.1	Question	691
5.11.2	Answer	691
<b>5.12</b>	<b>Performance Considerations</b>	<b>691</b>
<b>5.13</b>	<b>How to restore the structure using History</b>	<b>692</b>
<b>5.14</b>	<b>Can a Structure be Exported to Microsoft Word?</b>	<b>692</b>
<b>5.15</b>	<b>Convert time data in Excel export to Jira format</b>	<b>693</b>
5.15.1	How to Install	693
5.15.2	How to Use	693
<b>6</b>	<b>Structure Troubleshooting</b>	<b>695</b>
<b>6.1</b>	<b>Collecting Support Zip</b>	<b>695</b>
<b>6.2</b>	<b>HAR Network Report</b>	<b>696</b>
6.2.1	Collecting HAR Report with Google Chrome	696
<b>6.3</b>	<b>Troubleshooting Synchronizers</b>	<b>697</b>
6.3.1	Structure Audit Log	697
6.3.2	Log Files	697
<b>6.4</b>	<b>Structured JQL Troubleshooting</b>	<b>699</b>
<b>6.5</b>	<b>Collecting Performance Snapshots</b>	<b>699</b>
6.5.1	Download and install Atlas-Yourkit plugin.	699
6.5.2	Load Profiling Agent	700
6.5.3	Capturing CPU Performance Snapshot	700
6.5.4	Capturing Memory Snapshot	700
6.5.5	Sending the Snapshots to Support Team	701
6.5.6	After Profiling Session	701
6.5.7	Performance Snapshot Without Yourkit Plugin	701
<b>6.6</b>	<b>Sending Files to Support Team</b>	<b>706</b>

6.6.1	Attach to the Support Request in ALM Works Service Desk (Preferred)	707
6.6.2	Send Files by E-mail	707
6.6.3	Upload Files Directly to Our Server	707
<b>6.7</b>	<b>Troubleshooting Performance and Stability Issues</b>	<b>707</b>
6.7.1	Thread Dumps	708
6.7.2	Verbose Logging	708
6.7.3	Support Zip	709
6.7.4	Browser Console Log	709
6.7.5	HAR Report	709
6.7.6	Screenshots or Video	710

Links to the available documentation collections:

Download documentation:

Name	Version	Published
------	---------	-----------

# 1 Structure Quick Start Guide

Structure 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.

While we've done our best to make Structure as easy and intuitive as possible, we've created the following Getting Started guide to help you get the most out of your new tool. It 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.

## 1.1 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](#).

That's it! Now let's get started: [Creating Your First Structure](#)

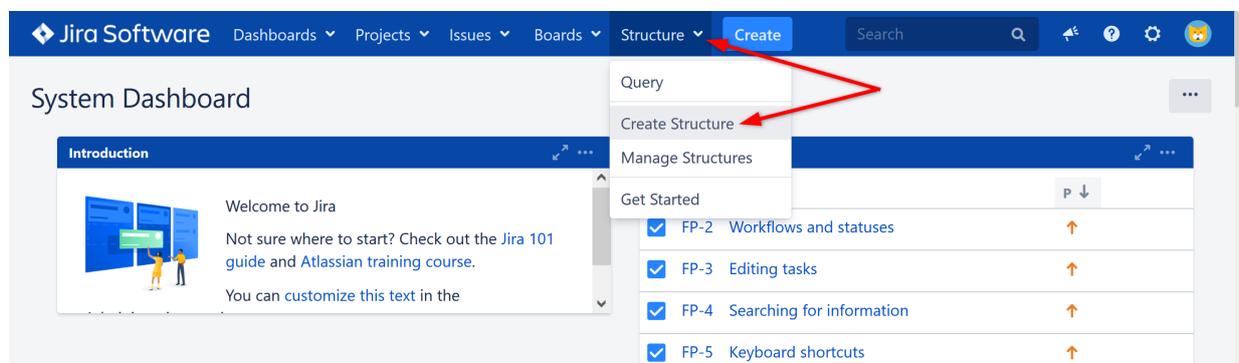
## 1.2 Creating Your First Structure

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](#) and steps for [adding issues manually](#).

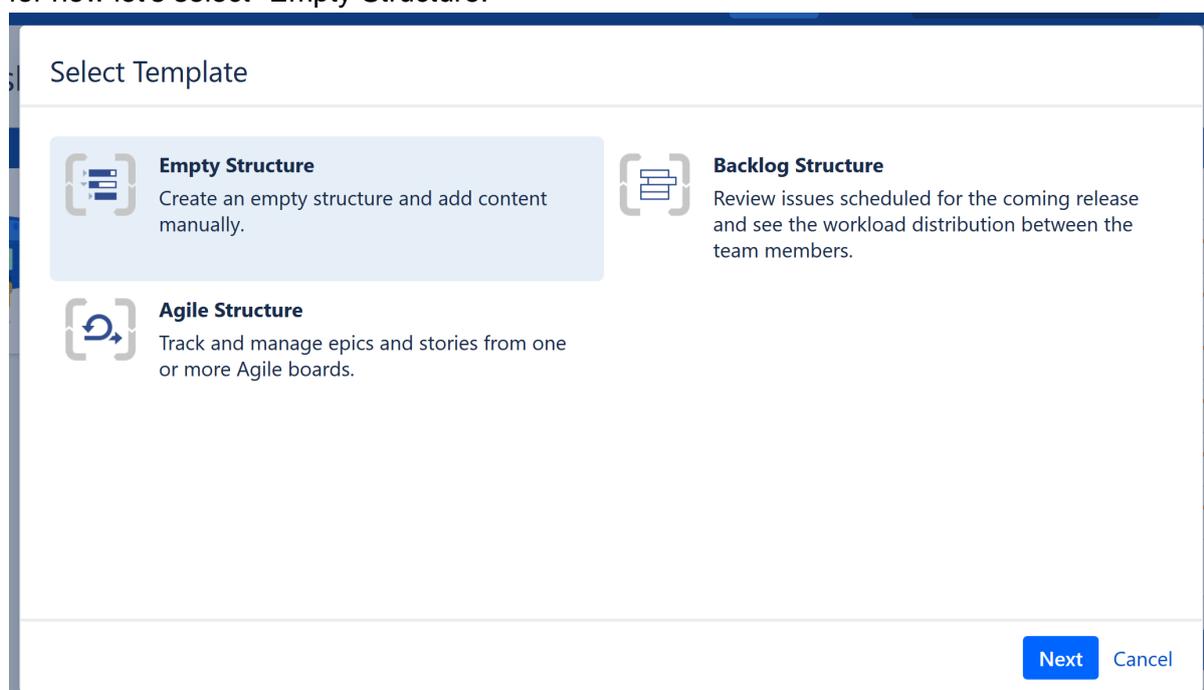
We encourage you to experiment with the tools presented here, make several new structures and find some templates that will work well for you and your team

## 1.2.1 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 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."



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

## 1.2.2 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!

### 1.2.3 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!).

#### A (Very) Brief Overview of Automation

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

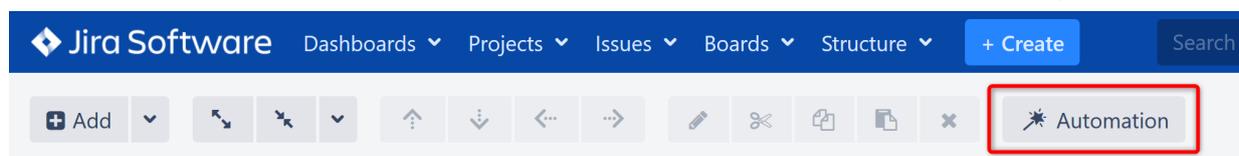
Automation uses **generators** - special rules that tell Structure what issues to show you from Jira and where to place them within your structure. Every time you open Structure, these generators will run again and completely rebuild your structure, based on the current information available. In this 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 [Automation](#).

For now, we'd like to show 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

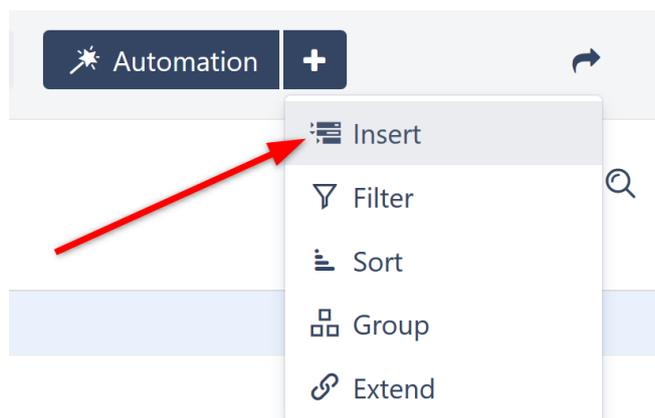
One of the great advantages to using Structure is the ability to manage issues from multiple projects in a single location. The following guide will walk you through one approach for adding Epics, Stories, and Tasks from multiple projects into a single structure.

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

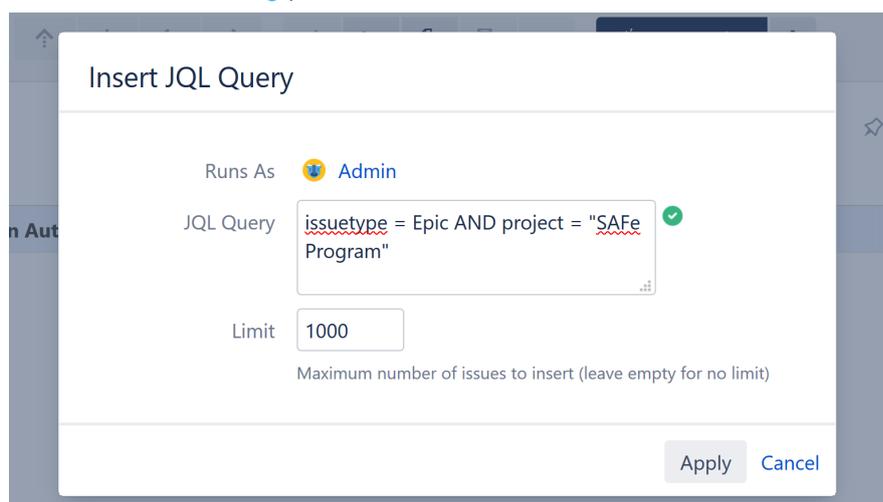


#### Step 1: Insert Epics

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](#).)



**i** 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.

Top-Down Automation

Key	Summary	Progress	Status	Assignee	Icons
+ Insert issues: <b>issuetype = Epic AND project = "SA"</b>					
SPR-12	SAFE Epic 12		BACKLOG	Unassigned	
SPR-11	SAFE Epic 11		IN PROGRESS	M. Reynolds	
SPR-10	SAFE Epic 10		BACKLOG	Unassigned	
SPR-9	SAFE Epic 9		SELECTED FOR DEVELOPMENT	Bob	

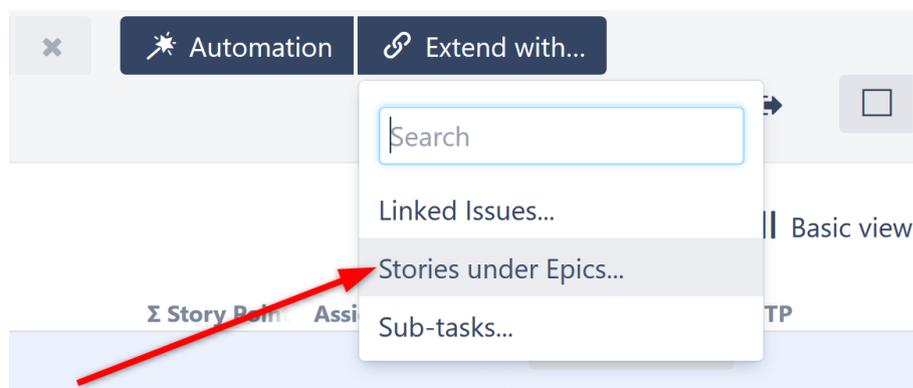
While you are in Automation Editing Mode, all generators are listed in red.

- 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.

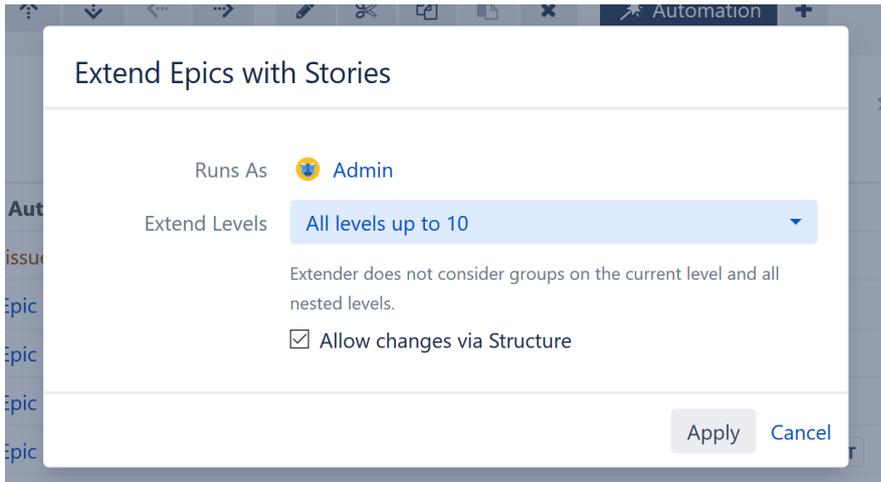
## Step 2: Extend with Stories

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**.



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.

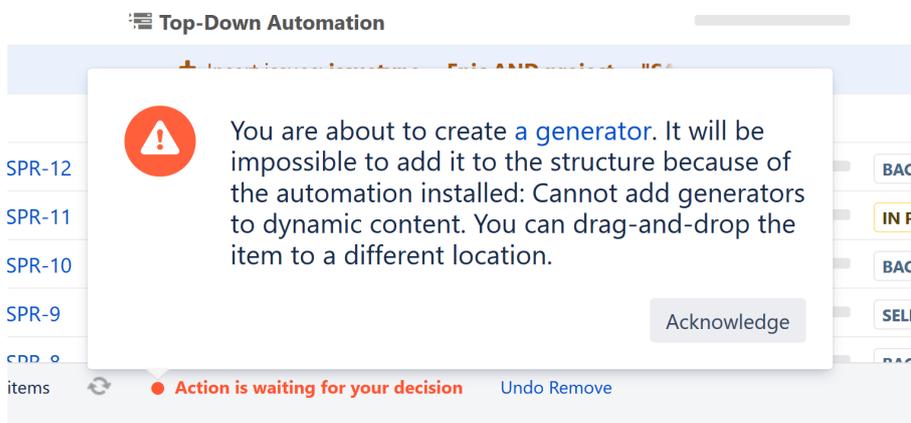


Click **Apply**. Some (maybe all) of your epics should now have a small arrow next to their Summary. Click the arrow to expand to the next level in the hierarchy – in this case, that's the Stories under each Epic.

Top-Down Automation ▾ 🔖 📁 🔍 📄 Basic view ▾

Key	Summary	Progress	Status	Assignee	Icons
SPR-12	SAFe Epic 12	<div style="width: 100%;"><div style="width: 100%;"></div></div>	BACKLOG	Unassigned	🔖 ⬆️
STMB-5	Team B Story 5	<div style="width: 100%;"><div style="width: 100%;"></div></div>	IN PROGRESS	Unassigned	👤 ⬆️
STMB-11	Team B Story 11	<div style="width: 100%;"><div style="width: 100%;"></div></div>	IN PROGRESS	Nah Duo	👤 ⬆️
SPR-11	SAFe Epic 11	<div style="width: 100%;"><div style="width: 100%;"></div></div>	IN PROGRESS	M. Reynolds	🔖 ⬆️
STMA-2	Team A Story 2	<div style="width: 100%;"><div style="width: 100%;"></div></div>	IN PROGRESS	Jack Brown	👤 ⬆️
STMA-1	Team A Story 1	<div style="width: 100%;"><div style="width: 100%;"></div></div>	IN PROGRESS	C. Bacca	👤 ⬆️

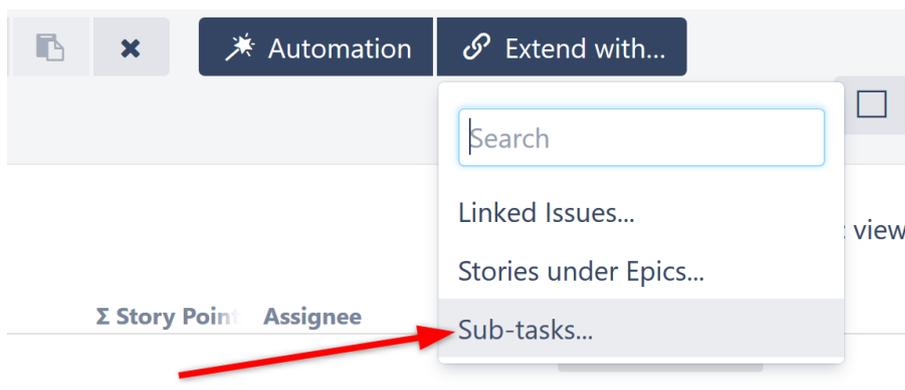
- ✔ If you forget to highlight the top line of your structure before adding the Extend generator, you may get the following error:



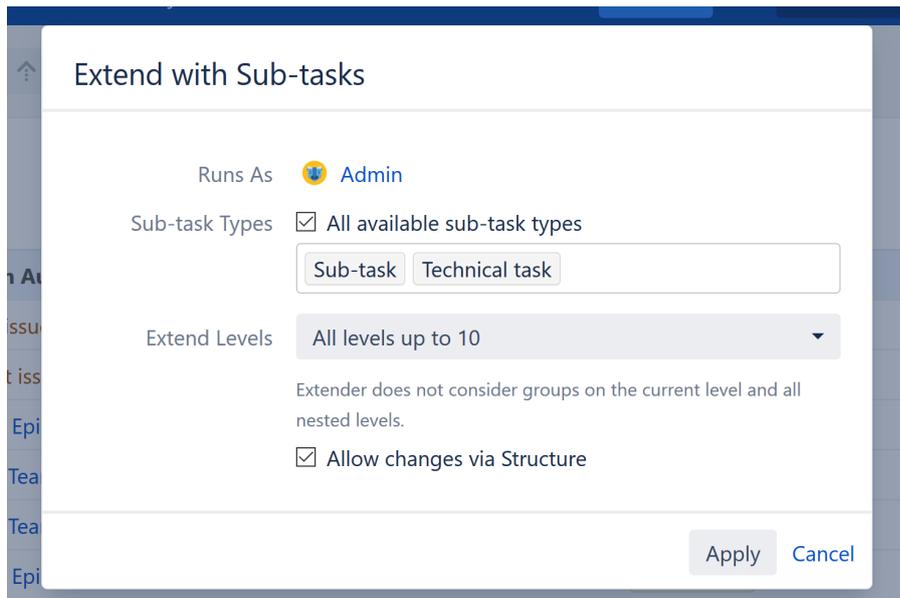
Since we started with a global generator (affecting the entire structure), all other generators need to be global as well, to avoid any conflicts. Simply highlight the top line of your structure (where it says the structure's name), and try again.

### 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](#).)

Top-Down Automation

Key	Summary	Progress	Status	Assignee	Icons
SPR-3	SAFe Epic 3	<div style="width: 50%;"></div>	IN PROGRESS	Bob	
STMB-10	Team B Story 10	<div style="width: 50%;"></div>	IN PROGRESS	Nah Duo	
✓ STMB-22	Sub-task 8	<div style="width: 100%;"></div>	DONE	Mary	
✓ STMB-23	Sub-task 9	<div style="width: 100%;"></div>	DONE	Mary	
STMB-24	Sub-task 10	<div style="width: 50%;"></div>	IN PROGRESS	Mary	
STMB-2	Team B Story 2	<div style="width: 50%;"></div>	IN PROGRESS	Mary	
✓ STMB-19	Sub-task 6	<div style="width: 100%;"></div>	DONE	Nah Duo	

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 [Automation](#) section of our Structure User's Guide.

**i** 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.

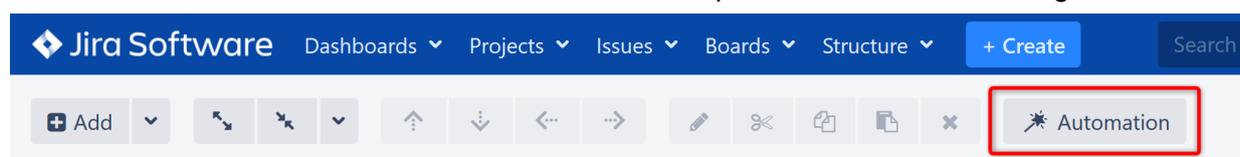
[Bottom Up Automation for Backlog Grooming](#)

## 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.

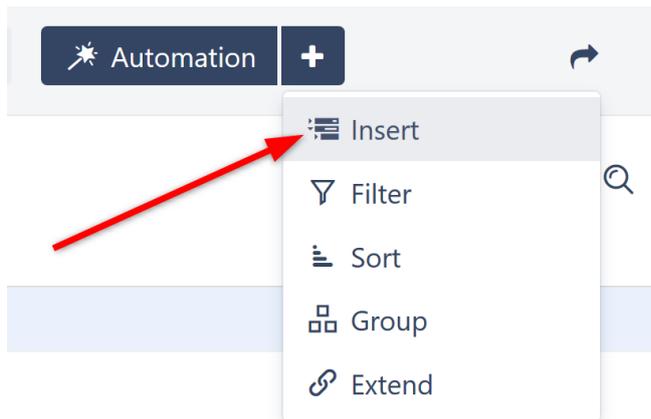
In the last tutorial, we started at the top of our hierarchy (Epics) and worked our way down. This time, we're going to start at the bottom and insert all of our active stories.

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

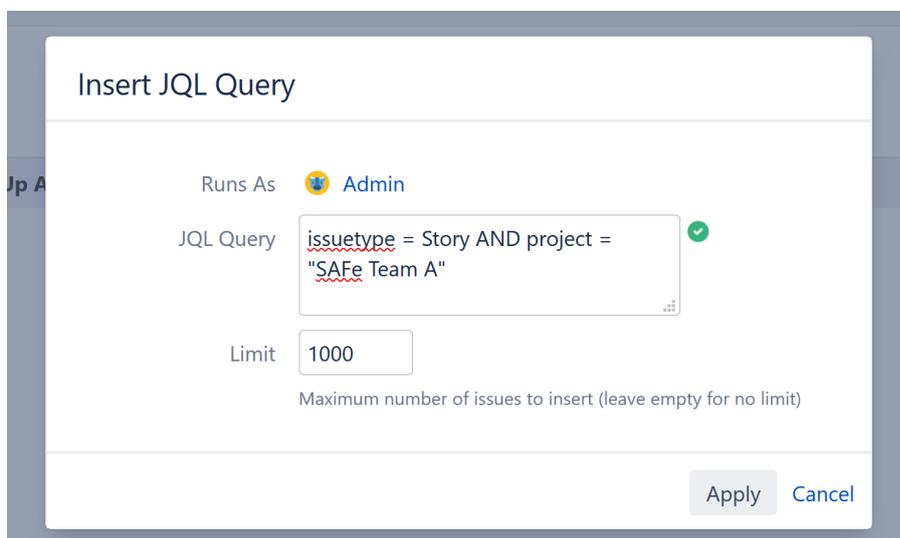


## Step 1: Insert Stories

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



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](#) for more information about using JQL).



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.

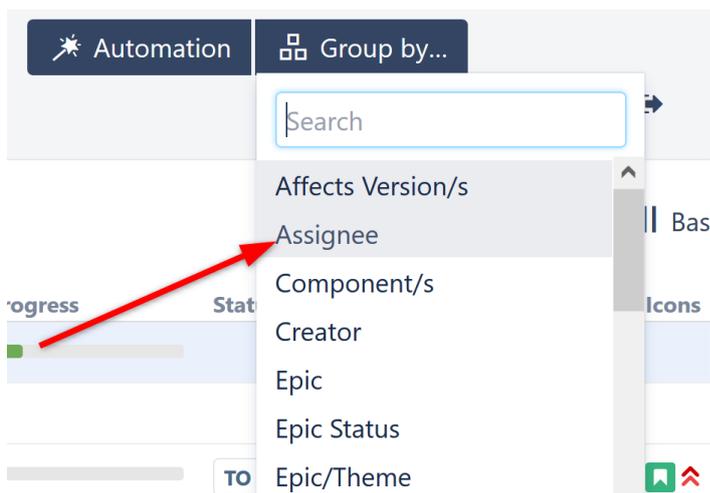
 A screenshot of a Jira issue list. The list is titled 'Bottom-Up Automation' and shows the following items:
 

Key	Summary	Progress	Status	Assignee	Icons
Bottom-Up Automation					
+ Insert issues: <code>issuetype = Story AND project = "SAFe Team A"</code>					
STMA-21	Team A Story 20	<div style="width: 100%;"></div>	TO DO	Unassigned	
STMA-16	Team A Story 16	<div style="width: 100%;"></div>	TO DO	Unassigned	
STMA-15	Team A Story 15	<div style="width: 100%;"></div>	TO DO	Unassigned	
STMA-14	Team A Story 14	<div style="width: 100%;"></div>	TO DO	Unassigned	
STMA-13	Team A Story 13	<div style="width: 100%;"></div>	TO DO	Unassigned	

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.

Bottom-Up Automation

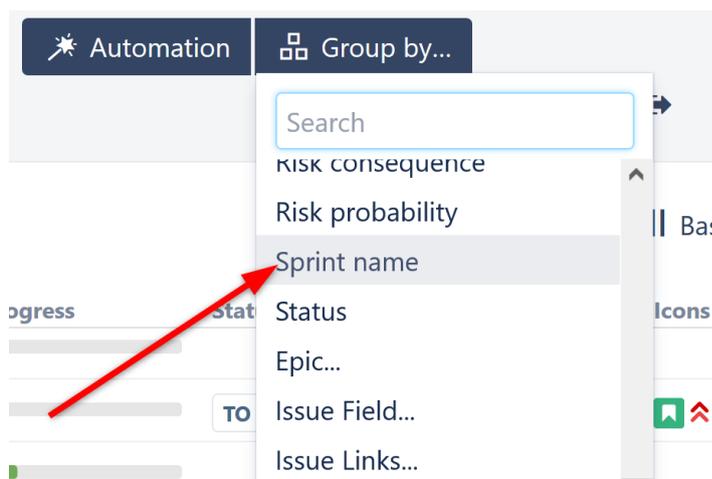
Basic view

Key	Summary	Progress	Status	Assignee	Icons
albert	Albert				
STMA-8	Team A Story 8		TO DO	Albert	
anna	Anna M.				
STMA-10	Team A Story 10		TO DO	Anna M.	
STMA-9	Team A Story 9		TO DO	Anna M.	
STMA-5	Team A Story 5		IN PROGRESS	Anna M.	

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.

Bottom-Up Automation ▾

☆ ☰ 🔍 Basic view ▾

Key	Summary	Progress	Status	Assignee	Icons
	Team A Sprint 1	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>			
anna	Anna M.	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>			
STMA-5	Team A Story 5	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>	IN PROGRESS	Anna M.	
STMA-4	Team A Story 4	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>	TO DO	Anna M.	
STMA-1	Team A Story 1	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>	IN PROGRESS	Anna M.	
claire	Claire T.	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>			
STMA-6	Team A Story 6	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>	IN PROGRESS	Claire T.	
jack	Jack Brown	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>			
	Team A Sprint 2	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>			
albert	Albert	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>			
STMA-8	Team A Story 8	<div style="width: 100%; height: 10px; background-color: #28a745;"></div>	TO DO	Albert	

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 [Automation](#).

## Next Steps

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

If you would rather keep things simple, jump ahead to [Working with Structure](#)

### 1.2.4 Building a Structure Manually

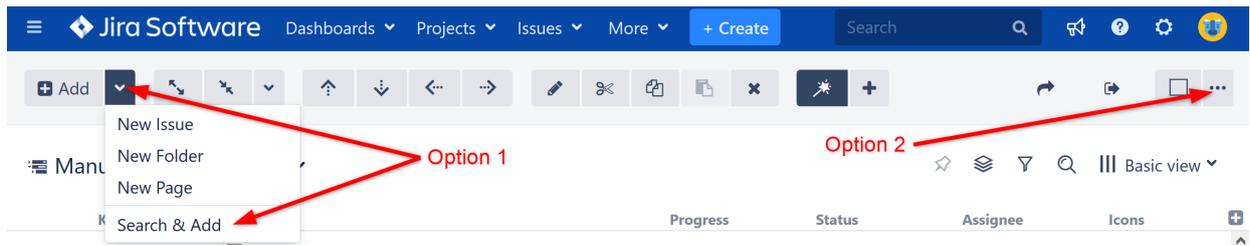
Building a structure manually allows you complete freedom in terms of choosing only the issues you want to include within your structure and arranging (or rearranging) them to your exact needs. When you add items to a structure manually, their position is stored within the Structure app itself – so you're free to move things anywhere you like, without affecting the issues in Jira.



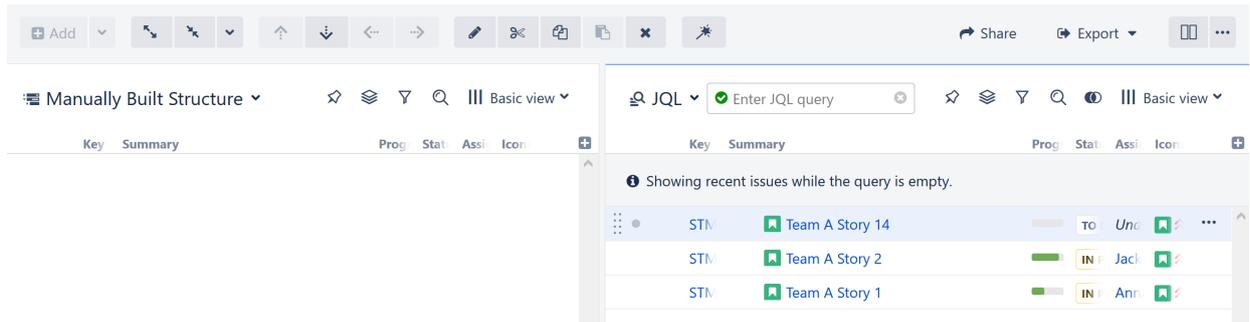
Changes to issue properties from within Structure are reflected in Jira, regardless of how your structure is built.

#### Step 1: Open the Secondary Panel

To manually add issues to your structure, use the **Add | Search & Add** toolbar button in the top left corner of the Structure Menu. Or you can get the same results by clicking the Layout control in the top right corner and choosing **Double Grid**.



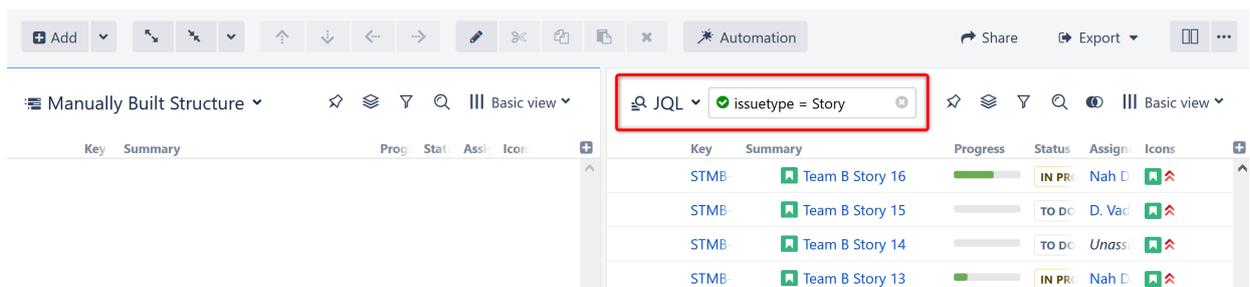
Either of these methods will open a second panel to the right of your structure. In this panel, you can enter a JQL query to search for issues you may want to add to your structure.



## Step 2: Search for Issues

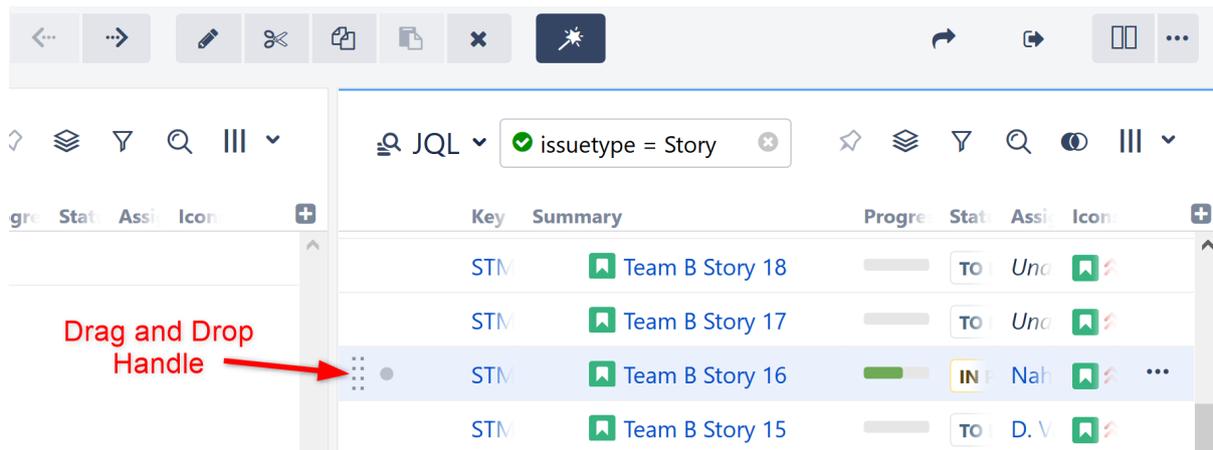
To find issues, enter a [JQL query](#) into the search box. (You can also search by plain text or add items from existing structures, but those are outside the scope of this tutorial – for more information, see [Adding Issues](#)).

Let's enter a simple query to find all of the Stories across your various projects. In the JQL entry field, enter **issuetype = Story**.

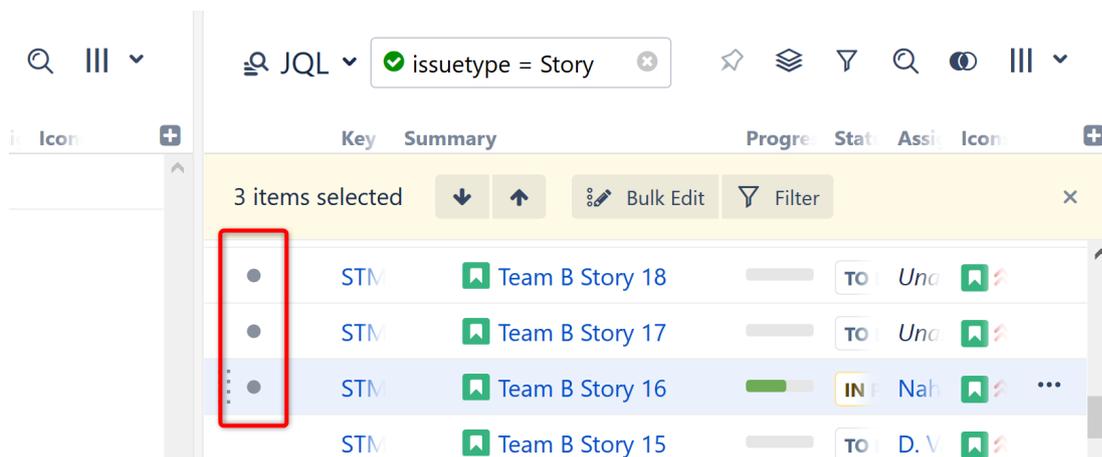


## Step 3: Add Issues to Your Structure

To add a single issue to your structure, use the drag and drop handle to the left of the issue (it only appears when you select or mouse-over an item), and drag the issue into your structure.



To add more than one issue at a time, click the small circle next to the drag and drop handle for each issue you want to add. Once you have selected all of the issues you want, use the drag and drop handle for any of your selected issues – this will allow you to drag all of them into your structure.



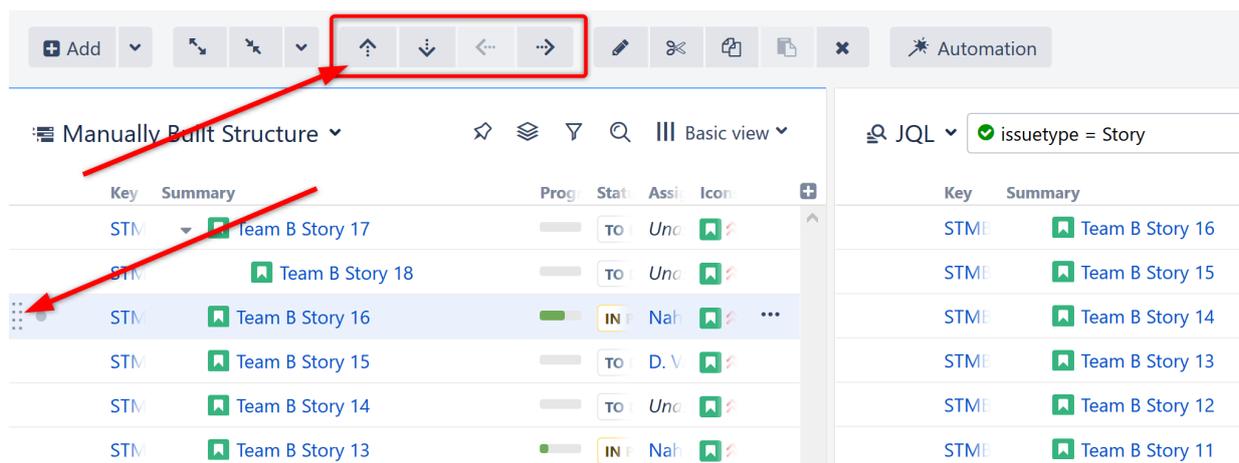
### Adding the Same Issue Multiple Times

Issues can appear more than once within a structure. For example, a single task might be a requirement for multiple stories - so you may want to include it beneath all of those stories.

To add an issue to multiple locations in a structure, simply drag the issue from the Secondary Panel again (or even several times). You can also copy an issue from within a structure, using drag and copy – just hold the **CTRL** key while you move the issue ( **Option** key on Macs).

### Step 4: Organize Your Issues

Once issues have been added to your structure, you can move them around or arrange them into a hierarchy using [drag-and-drop](#) or the control buttons in the [toolbar](#) above the structure board.



We'll discuss more about moving issues in [Working with Structure](#).

Congratulations! You've created your first structure. Now let's learn how to make dynamic structures that will unleash the full power of your newest productivity tool.

## Next Steps

Now that you've learned three different ways to create new structures, let's take a look at what you can do with those structures.

### [Working with Structure](#)

## 1.3 Working with Structure

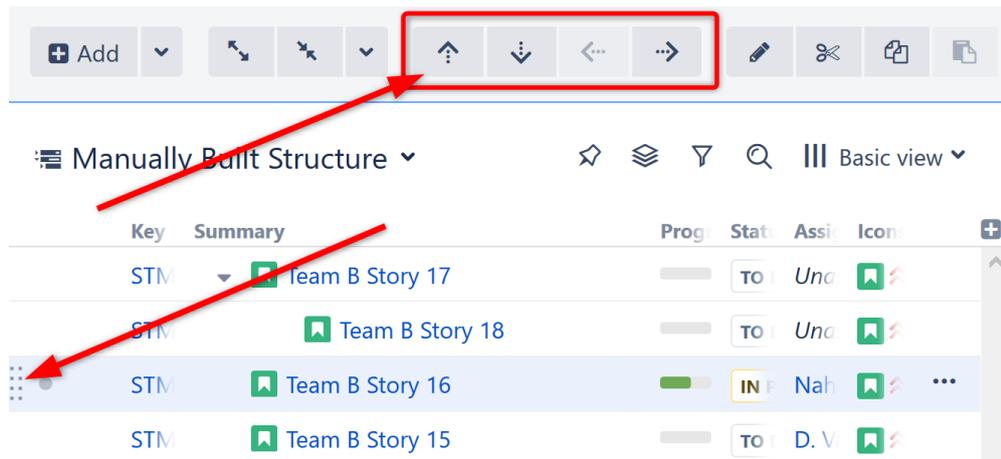
### 1.3.1 Moving, Adding and Removing Items from a Structure

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

#### 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**.



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

## 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](#)), or
- Use Automation (see [Building a Structure with Automation](#)).

## 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](#)). Once you have selected all the items you want to remove, click the delete key in the toolbar or use your keyboard.

Manually Built Structure

3 items selected

Key	Summary	Progress	Status	Assignee	Icons
STM	Team B Story 18		TO DO	Unassigned	
STM	Team B Story 17		TO DO	Unassigned	
STM	Team B Story 16		IN PROGRESS	Nah Duo	
STM	Team B Story 15		TO DO	D. Vader	



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

### 1.3.2 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.

Progress Status Assignee Icons

Horizontal scrolling

Search

- FEATURED
- BugFix % (Time)
- WSJF (Basic)
- PROGRESS
- Custom Progress...
- Progress by Original story points

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

### Special Columns and Views

Structure also offers several very useful custom columns, including:

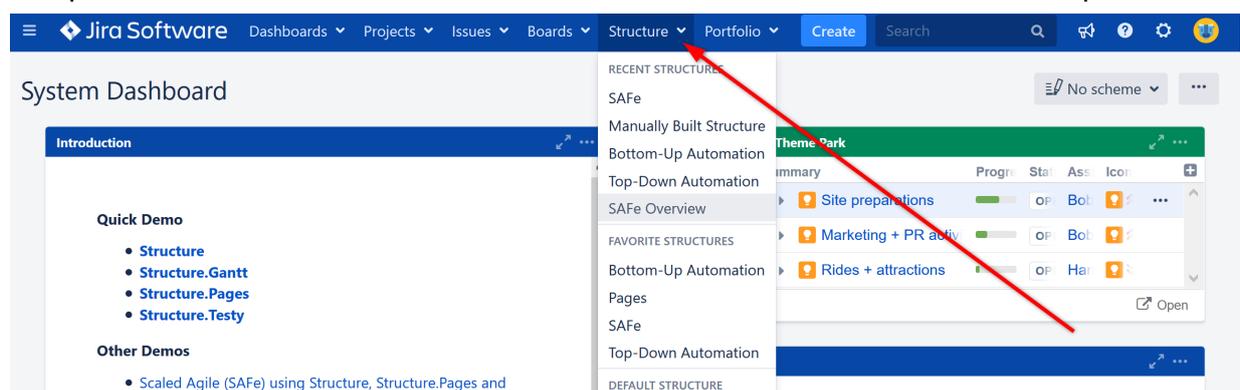
- **Progress** - Progress can be calculated based on the **Time Tracking**, **Status** or a custom field such as **Story Points**. To review (or change) your Progress calculation settings, hover over the Progress title in the column header and click the small triangle. You can find all the details on how the progress is calculated in the [full documentation](#).
- **Work Logged** - The Work Logged column displays the work logged for every issue over a selected period.
- **Totals** - 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** - 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.

### 1.3.3 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.

### 1.3.4 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](#).

### 1.3.5 Next Step

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

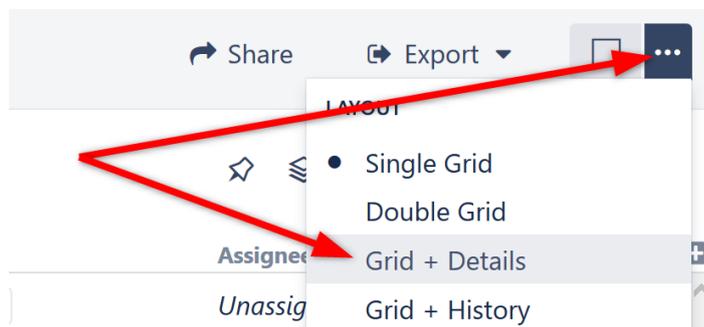
#### Working with Issues in Structure

### 1.3.6 Working with Issues in Structure

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

#### 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.

The screenshot shows the Jira Structure interface. On the left, a table lists issues under 'Manually Built Structure'. On the right, the 'Issue Details' panel is open for 'Team B Story 14'.

Summary	WSJF (Basic)	Progress
Team B Story 18	677	
Team B Story 17	342	
Team B Story 16	332	
Team B Story 15	291	
Team B Story 14	317	
Team B Story 13	371	
Team B Story 12	618	
Team B Story 11	349	

**Issue Details Panel:**

- Issue:** Team B Story 14
- Type:** Story
- Status:** TO DO
- Priority:** Major
- Resolution:** Unresolved
- Affects Version/s:** None
- Fix Version/s:** 2.0 - Team B
- Assignee:** Unassigned
- Reporter:** M. Reynolds
- Votes:** 0

## 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](#).

TP	Assignee	Status	WSJF (Basic)
	Unassigned	BACKLOG	489
	<input type="text" value="Jack Brown"/>	Done	718
	Nah Duo	IN PROGRESS	349
	Jack Brown	IN PROGRESS	363

If the field is not visible, you can [add the column](#) 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](#), [view and add comments](#), [share](#), [view history](#), [view development information](#), and more.

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

### Jira Actions Column

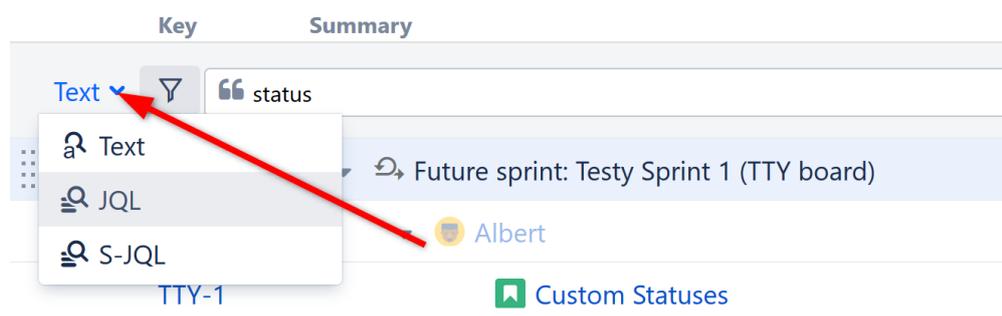
Using the Jira Actions Column (far right column of the structure board), you can conduct many of the same actions available within the Issue Details Panel directly from you structure. This column works like the similar column on the Jira Issue Navigator page and lets you log work, apply workflow actions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



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.

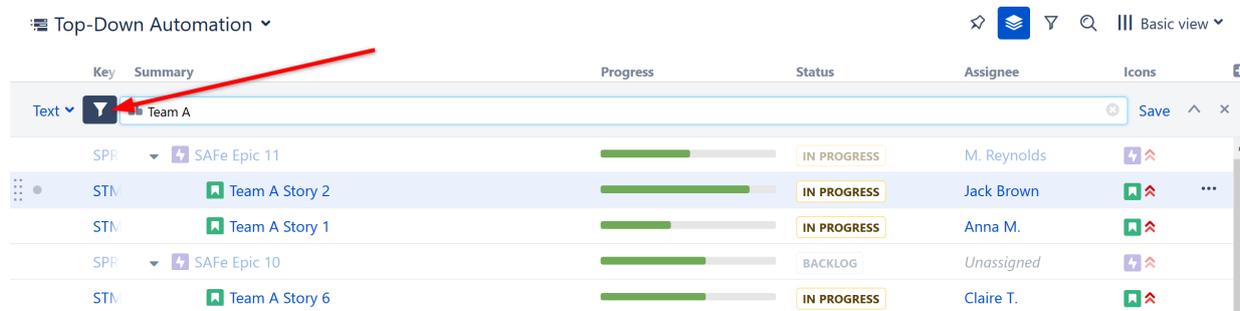
## Search Options

You can [search for issues](#) 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.



## 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.



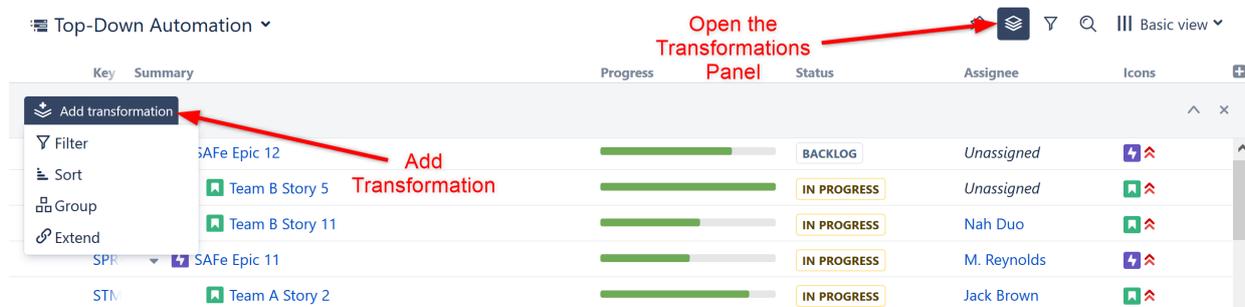
## Transformations

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



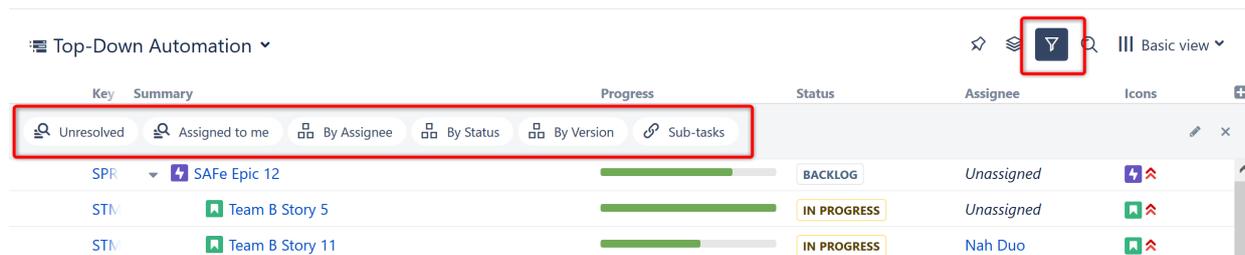
When you are finished with the transformation, simply click the **x** on the right side of the Transformations Panel and the transformation will be removed.

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

## 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](#).*

## 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](#)

## 1.4 Help and Support

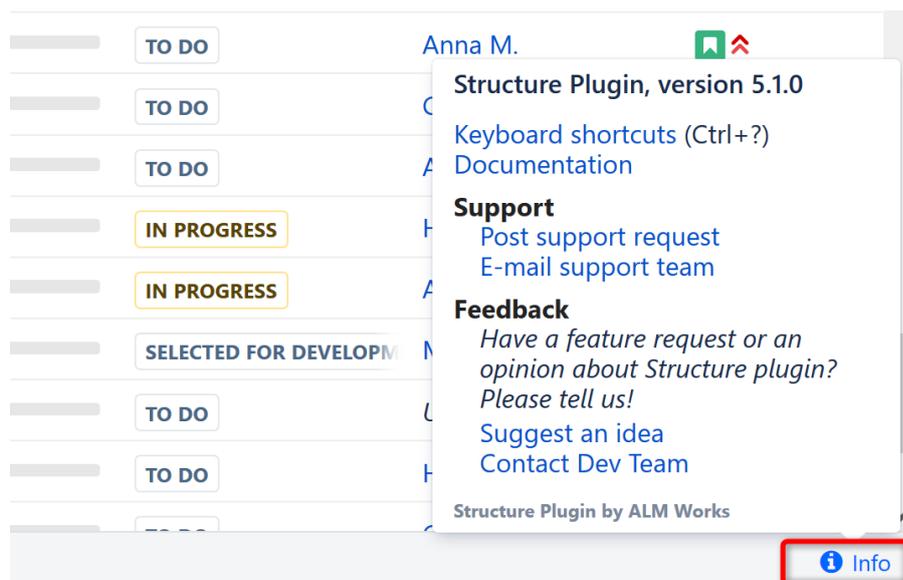
You have now seen how easy it is to create and work with structures – and hopefully you're excited to try creating some new structures all on your own. Good luck and have fun!

### 1.4.1 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](#) or contact our support team.

### 1.4.2 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 a list of keyboard shortcuts, our resource documentation, and our support services.



## 1.5 Structure.Pages Quick Start Guide

### 1.5.1 Basic Facts

Structure.Pages is an extension for Structure plugin that lets you create, manage and organise Confluence pages alongside JIRA issues right inside Structure.

Once you connect your JIRA and Confluence with Application Links and configure Structure. Pages you'll be able work with pages from your Confluence in a similar way you work with issues:

- Find pages that exist in your Confluence using text search or CQL and add them to your structure
- View and edit pages right from the Structure interface in the details panel
- Create new pages right inside the structure
- Organise pages into hierarchy in your structure manually or automatically based on your existing Confluence hierarchy
- Automatically pull in Confluence pages linked to issues in your structure

### 1.5.2 Installation

Structure.Pages provides integration between JIRA and Confluence so its installation requires a bit more work than installation of a plugin that only works with one system. But fear not - follow the steps described below and you'll have it running in no time. And if something goes wrong, please don't hesitate to contact us at [support@almworks.com](mailto:support@almworks.com).

Here's what you'll need to do:

1. Configure Application Links between your JIRA and Confluence
2. Install the special Confluence helper add-on
3. Install the JIRA add-on and configure the integration

## 1. Setting up Application Links

To make your JIRA and Confluence work together and exchange data you need to set up Application Links. You can find all the details on how they work and how to set them up in [Atlassian documentation](#).

Atlassian recommends using [OAuth](#) protocol because of the greater security provided by that protocol and Structure.Pages supports this and other available protocols.

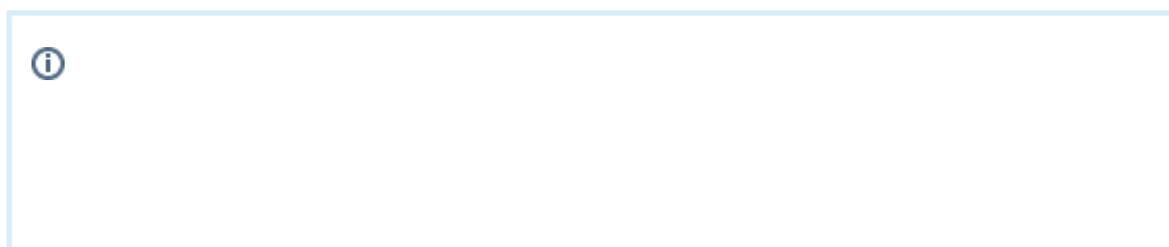
## 2. Installing Pages Add-On for Confluence and Configuring Confluence

To allow Structure communicate with Confluence you need to [install a small add-on for Confluence](#). This add-on does not require any additional configuration and once installed it will take care of the integration with Structure on the Confluence side. You will also need to enable **Remote API (XML-RPC & SOAP)** in Confluence. Use menu **Administration | Further Configuration**. Structure.Pages needs XML-RPC to access methods not available through the REST API.

## 3. Installing and Configuring JIRA Add-On

JIRA add-on for Structure.Pages requires a bit more work:

1. Install [Pages add-on for JIRA](#) the same way you installed the add-on for confluence.
2. Once the add-on is installed go to **Administration | Structure** page and select **Confluence Integration** in the menu on the left.
3. You'll see the list of Confluence instances you can connect to. These are the instances for which you have configured Application Links.
4. Click **Configure Integration** next to the Confluence you want to work with.
5. Select the **System Authentication** method. The drop-down shows all authentication types you have configured in the Application Links.



It's recommended to choose the one that would allow Structure.Pages see the most of the Confluence. All users who will work with Confluence via Structure.Pages will see a sub-set of pages that System Authentication permits to see, so if System Authentication permissions are more restrictive than the user permissions, the user won't be able to see some pages that should actually be visible.

OAuth method is preferred here and it's recommended to use admin user for this authentication.

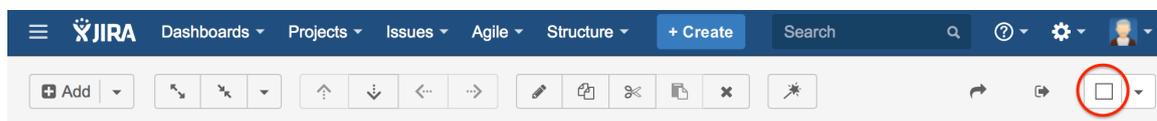
6. Select the **User Authentication** method. These settings are used when a specific user is working with the Confluence through Structure: looking at the search results, creating or editing a page, changing links between issues and pages. Structure.Pages offers authentication methods in order of preference, so the default one is usually the best.
7. Click **Verify & Save Settings** to apply the selected options. If you haven't enabled Remote API in Confluence when you were installing the helper add-on, you'll get a message with instructions how to do that. Once it's enabled click **Verify & Save Settings** again to save the settings.
8. Once you are done with the configuration you can start working with Confluence in Structure.

### 1.5.3 Main Functionality

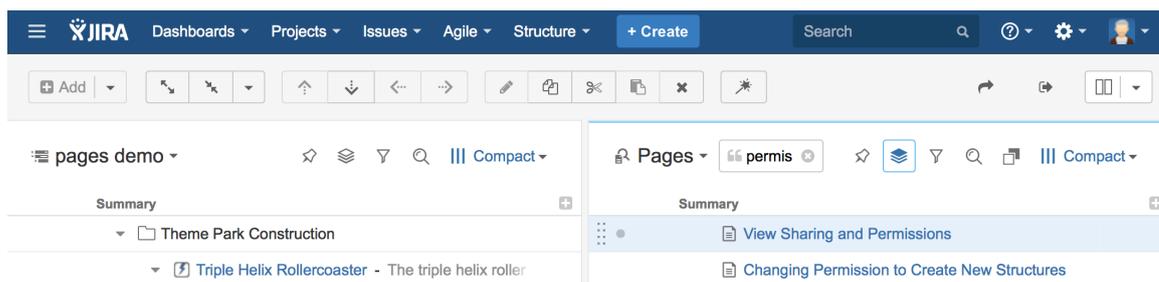
#### Find Existing Pages and Add Them to Your Structure

The simplest way to get started is to add some of the existing pages to your structure:

1. Open the structure, which you want to add pages to.
2. Open the second panel by clicking the panels toggle button.



3. In the second panel you will see either another Structure or **JQL** or **Text** search. Click the structure name or the **JQL/Text** label next to the search field and select **Confluence Pages** from the menu.
4. You will see the search field, which you can use to search for pages in your Confluence. The search works just like the search field in Confluence itself and you can use the exact same [syntax](#).
5. As you type, the results will be shown in the panel.



6. Once you have found the pages you need, select them in the search results and add them to the structure using drag-and-drop.

## View and Edit Pages

Once the page is in a structure, you can see the page contents and edit it.

To do that simply click this page name in your structure and it will be shown in the panel on the right. You can work with it as if it was a usual Confluence page opened in a browser. You can click **Edit** to start editing the page contents or use any other available controls.

**i** If you are not logged into the Confluence in the same browser session, you will be prompted to log in. Specify your Confluence **Username** and **Password** and click **Log in**.

**i** Next to the panels toggle button there is a down arrow that opens the menu, where you can select the layout and the action that happens when you click an issue or a page in a structure. If you have **Do Nothing** option selected, the second panel will not be opened as you click the items in your structure.

## Create New Pages

You can create new pages in your Confluence right from the Structure page:

1. Select an item in your structure, under which the page should be added.
2. Press Enter to create a page on the same level, or Shift+Enter to create it on the level below. You can also use the **Add** button in the top left corner.
3. As you press it, the **New Item** panel is shown. The panel allows to create issues, pages and folders. Switch to the **Page** tab.

**i**

If you've been using the full **New Issue** dialog to create issues, this dialog will be shown as you press Enter. Click the **Switch to panel** link to switch to the **New Item** panel.

4. Specify the page name and select the Confluence Space where the page should be created.
5. Click **Done** to create the page.

## Add Automation

You can configure some automation rules for pages in your structure in a similar way you do for issues.

## Child Pages

You can pull in all child pages of the pages previously added to your structure. To do that you need to switch on the **Automation** editing mode, select the top-most item that appears at the top of the Structure and then add the **Child Pages Extender** from the **Automation** menu.

-  As you move a child page from one parent page to another in your structure, this page in Confluence will also be moved from one parent page to another. If you want to disable such updates, switch on **Automation** editing, double-click the **Child Pages Extender** and select the option for disabling changes via Structure.

## Pages Linked to Issues

If you have links between pages and issues, you can visualise them in Structure too. To pull in pages linked to issues in your structure, add the **Linked Pages Extender** to your structure. Currently you cannot automatically pull in issues linked to pages in a structure, but this will be added in the future versions.

-  Moving pages between issues in Structure will update the links. If this undesirable, disable editing via Structure in the **Linked Pages Extender** settings.

## 2 Structure User's Guide

If you're new to Structure, we suggest you begin by reviewing our [Getting Started Guide](#).

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

[Expand all](#) [Collapse all](#)

### 2.1 Navigating Structure

#### 2.1.1 Basic Concepts

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

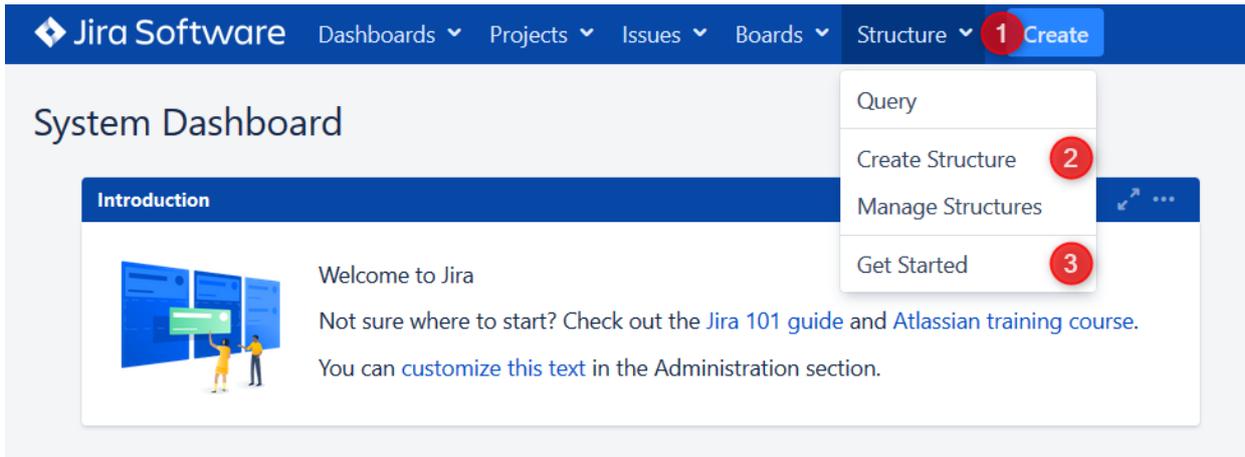
<b>Structure (vs. structure)</b>	<i>Structure</i> is the name of our product. In our documentation we differentiate between <b>Structure</b> , the app, and the <b>structures</b> that you build with it using capitalization. When you see “ <i>Structure</i> ” with a capital “ <i>S</i> ” we are referring to the app. When you see “ <i>structure</i> ” with a lowercase “ <i>s</i> ” we are referring to the the structures you create in the app.
<b>structures contain Jira issues</b>	Think of a structure as a container that may be filled with Jira <b>issues</b> 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.
<b>items</b>	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 <b>items</b> .
<b>automation / generators</b>	As mentioned above, Jira issues may appear in a structure automatically. Our powerful <b>Automation</b> feature uses <b>generators</b> 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.

<b>a view</b>	We refer to a particular configuration of the columns that you decide to display in the Structure panel as a <i>view</i> .
<b>sub-items (and parent items)</b>	<p>When you place one item under another item in a structure it becomes a <i>sub-item</i> of the item above it. The item above the sub-item is the <i>parent</i> 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.</p> <p>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.</p>
<b>children</b>	Sometimes we refer to sub-items as <i>children</i> (of the parent item).
<b>Jira sub-tasks</b>	<p>Jira <i>sub-tasks</i> and Structure <i>sub-items</i> 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.</p> <p>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.</p>
<b>structures within structures</b>	With Structure, <i>you may add structures to other structures</i> — i.e., a structure can be an item (see above) in another structure.

### 2.1.2 Structure Menu

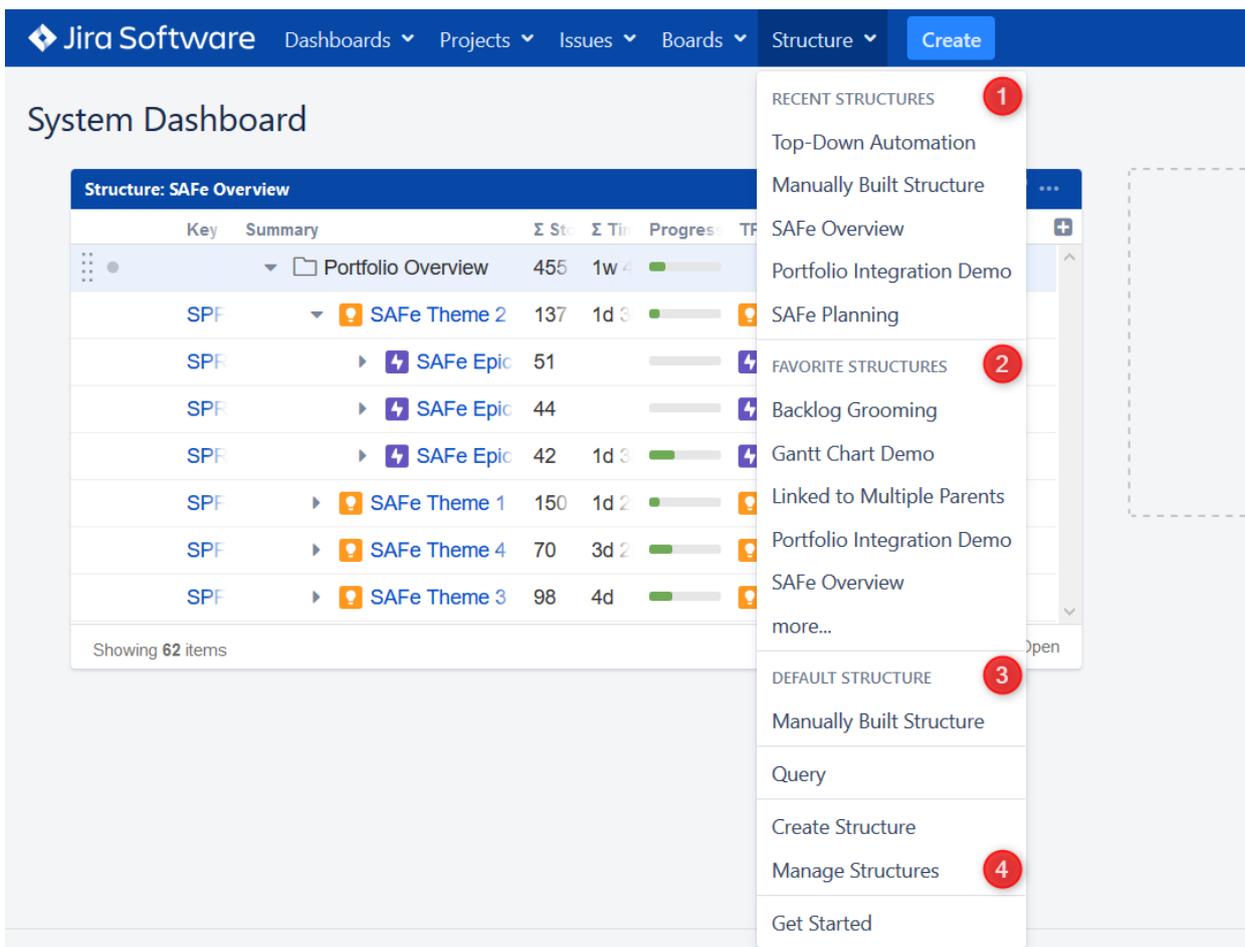
Once you install Structure, you will see a new item added to the top-level navigation bar (1).

If you haven't created any structures yet (or had structures shared with you), the menu will allow you to create a new structure (2) or view our Getting Started guide (3).



**i** Haven't read our Getting Started guide? It's the best way to get up to speed fast and take full advantage of the powerful functionality Structure has to offer.

Once you create your first structures (or are given access to existing structures), the Structure menu will also provide quick access to your favorite and recent structures:

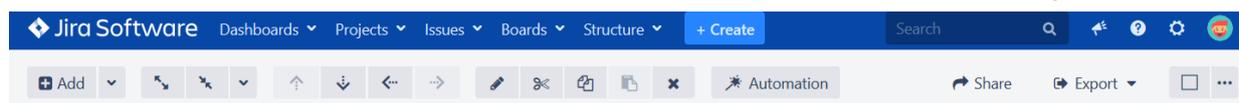


The menu has several sections:

1. **Recent Structures.** Shows structures that you've visited recently, or those which have been recently updated. Click a structure's name to open it.
2. **Favorite Structures.** Lists structures you have marked as your favorite. This section will be hidden if you don't have any favorite structures.
3. **Default Structure.** Shows the system-wide [Default Structure](#). If a separate default structure is [defined](#) for the current project, it will be listed here as well.
4. **Manage Structures.** Takes you to the [Manage Structures](#) screen, where you can view all of 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).

### 2.1.3 Main Structure Toolbar

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

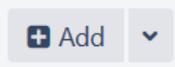


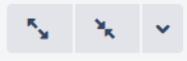
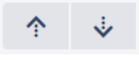
As you move the mouse pointer over the buttons in the toolbar, the active buttons are highlighted. Some buttons in the toolbar may be grayed out, indicating that these actions are not currently possible. For example, you cannot use the Paste action unless you have items in your clipboard, so the button 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.

### Available Actions

The following actions are accessible from the Structure toolbar.

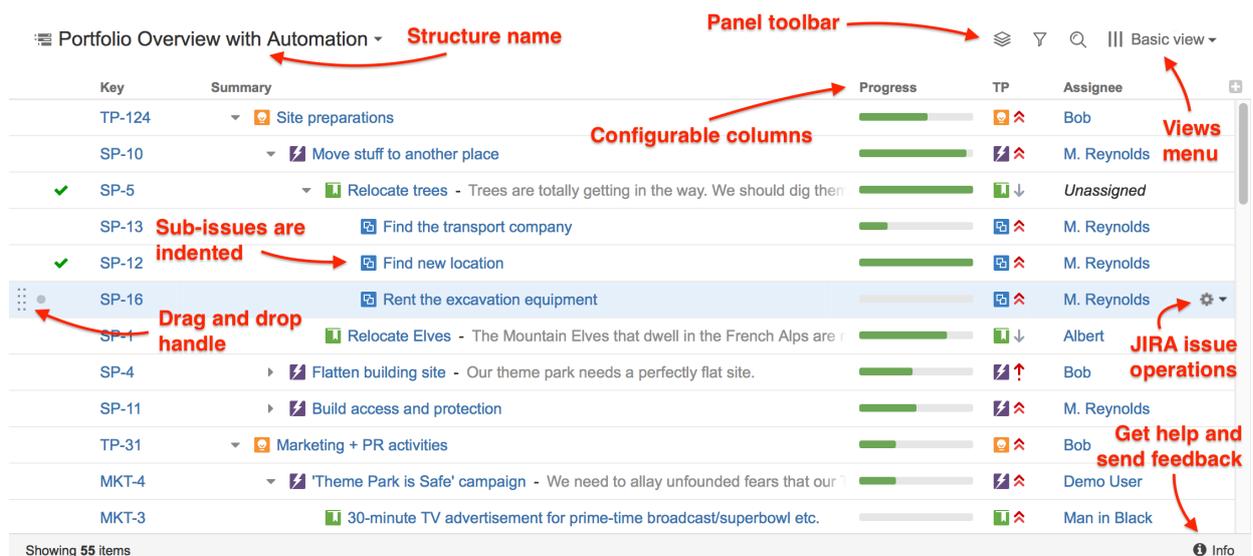
Button	Action	More Information	Keyboard Shortcut
		Creating new items	<b>Enter</b>

Button	Action	More Information	Keyboard Shortcut
	<p>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).</p> <p>To search for existing issues, click the drop-down menu next to the Add button.</p>		
	Expand or collapse the whole hierarchy. Use the drop-down menu to expand the hierarchy to a specific level.	<a href="#">Navigating Structure</a>	<b>++ / --</b>
	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.	<a href="#">Moving Issues Within Structure</a>	<b>Ctrl+Up / Ctrl+Down</b> or <b>Command+Up /Down</b>
	Unindent / Indent the item one level. If either or both of these moves are not possible, the option(s) will be grayed out.	<a href="#">Moving Issues Within Structure</a>	<b>Ctrl+Left / Ctrl+Right</b> or <b>Command+Left/Right</b>
	Edit the currently selected issue / stop editing and save changes.	<a href="#">Editing Issues</a>	<b>Tab</b>
	Cut the selected item(s) to the <a href="#">clipboard</a> .	<a href="#">Issue Clipboard</a>	<b>Ctrl+x</b> or <b>Command+x</b>
	Copy the selected item(s) to the <a href="#">clipboard</a> .	<a href="#">Issue Clipboard</a>	<b>Ctrl+c</b> or <b>Command+c</b>
	Paste item(s) from the <a href="#">clipboard</a> into the structure.	<a href="#">Issue Clipboard</a>	<b>Ctrl+v</b> or <b>Command+v</b>
			<b>Delete</b>

Button	Action	More Information	Keyboard Shortcut
	Remove the currently selected item(s) from the structure.	<a href="#">Removing Issues</a>	
	Switch Automation editing mode on/off.	<a href="#">Automation</a>	~
	Create a <a href="#">perspective</a> link to share the current view.	<a href="#">Perspective</a>	
	Open a <a href="#">printable page</a> with the structure or <a href="#">export the structure to Excel</a> .	<a href="#">Printing</a> <a href="#">Exporting to Excel</a>	
	Select the layout options, including opening the <a href="#">secondary panel</a> .	<a href="#">Secondary Panel</a>	

### 2.1.4 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](#) 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](#)

To learn more about performing each of these actions, see [Moving Items within Structure](#) and [Working with Issues](#).

 Structure widget is displayed on the [Structure Board](#), [Issue Page](#) and in [other places in JIRA](#).

## Navigating Between Items

### Navigating with Mouse

Using your mouse, you can:

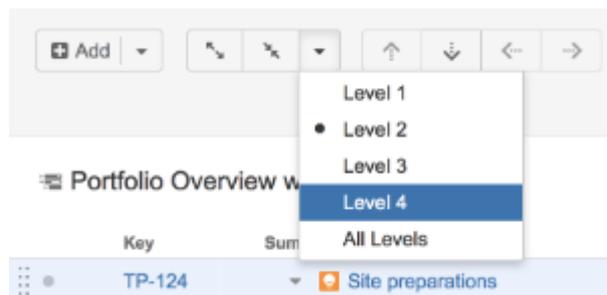
- Select an item by clicking anywhere in the item's row, except on a link
- Scroll up and down
- View the [Issue Details](#) panel by clicking the Key or Summary link (to customize this behavior, see [Viewing Issue Details](#))
- Show or hide sub-items

### Show/Hide Sub-Items

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

TP-124	Site preparations		Bob
SP-10	Move stuff to another place		M. Reynolds
SP-4	Flatten building site - Our theme park needs a perfectly flat		Bob
SP-11	Build access and protection		M. Reynolds

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.



**i** 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](#) available to simplify your Structure experience. Press **Ctrl+?** to view the shortcuts cheat sheet, or click **Info** at the bottom of the structure widget.

## 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.

Portfolio Overview with Automation ▾

5 items selected [Bulk Edit] [Filter]

Key	Summary	Progress	TP	Assignee
OFW-1	Overunity Ferris Wheel - @albert we'll need to change the l	<div style="width: 100%;"></div>	⚡🚫	Bob
OFW-2	Change the laws of thermodynamics in our favour - The	<div style="width: 100%;"></div>	📌📌	Albert
DTD-1	Dumper truck dodgems - Dodgem car ride with 60-tonne du	<div style="width: 100%;"></div>	⚡🔥	Mary
DTD-2	Tune up dumper truck engines	<div style="width: 100%;"></div>	📌🔥	Mary
DTD-3	Reinforced boundary walls - We need heavily reinforce	<div style="width: 100%;"></div>	📌🔥	Mary
LHMGR-2	Large Hadron Merry-go-Round - Merry-go-round based on p	<div style="width: 100%;"></div>	⚡🔥	Albert
LHMGR-3	Arrange black hole insurance	<div style="width: 100%;"></div>	📌🔥	Bob
LHMGR-4	Redefine the Standard Model of particle physics - It wo	<div style="width: 100%;"></div>	📌🔥	Bob

The selection panel offers the following features:

- Move focus from one selected item to another by clicking the up and down arrows
- **Bulk Edit** 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):

OFW-1	Overunity Ferris Wheel - @albert we'll need to change the l	<div style="width: 100%;"></div>	⚡🚫	Bob
DTD-1	Dumper truck dodgems - Dodgem car ride with 60-tonne du	<div style="width: 100%;"></div>	⚡🔥	Mary
LHMGR-2	Large Hadron Merry-go-Round - Merry-go-round based on p	<div style="width: 100%;"></div>	⚡🔥	Albert

The meaning of each marker is as follows:

	The item and <b>all</b> its sub-items are selected.
	The item and <b>some</b> of its sub-items are selected.

	The item itself <b>is not</b> selected, but <b>some</b> of its sub-items are selected.
	The item itself <b>is not</b> selected, but <b>all</b> of its sub-items are selected.

## Changing Multiple Items

The following actions work with the multi-selection:

- [Drag and drop](#) lets you move a selection of items within a structure or between two structures
- [Cut and paste](#) allows you to move items within a structure and between different structures
- [Remove button](#) 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](#) lets you use the Jira bulk change wizard to modify the selected issues

## Exiting Multi-Select Mode

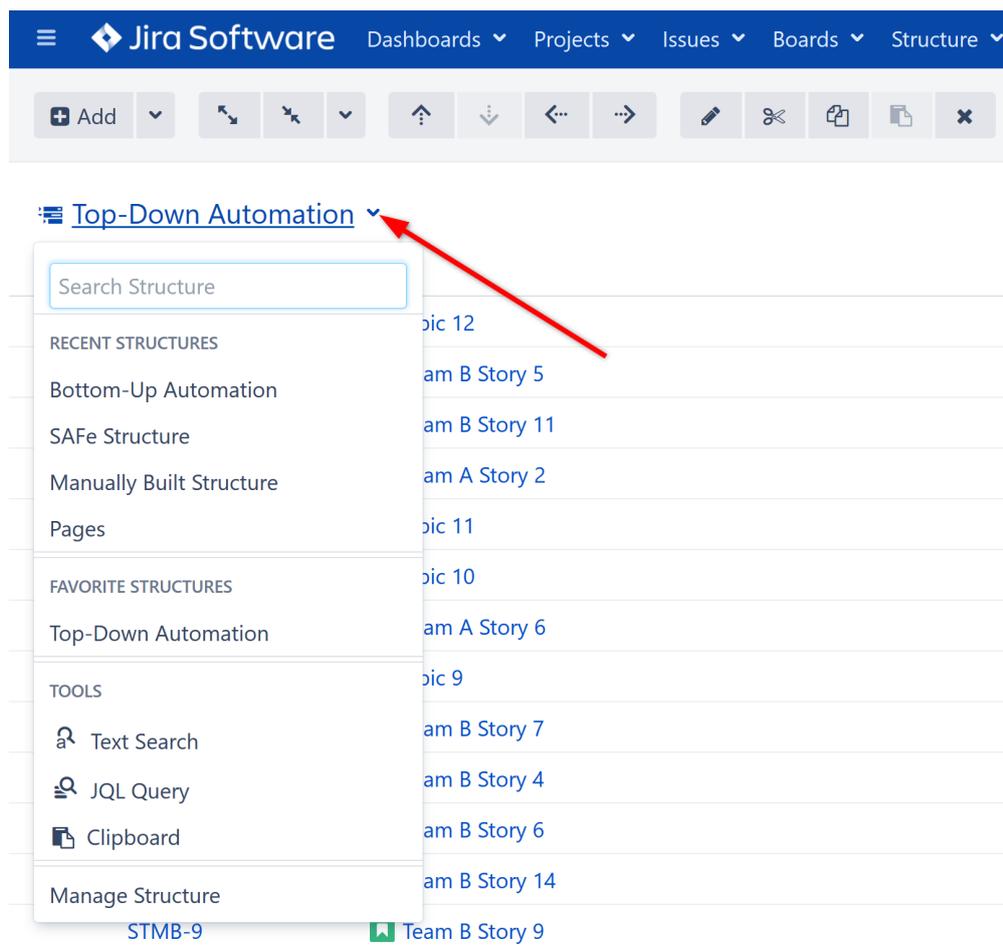
To exit multi-select mode (and deselect all items), press the **x** button at the far right of the selection panel or press the **Escape** key.



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

## Switching Between Structures

If you need to switch structures without leaving the current Structure widget, instead of using the [Structure menu](#), 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.

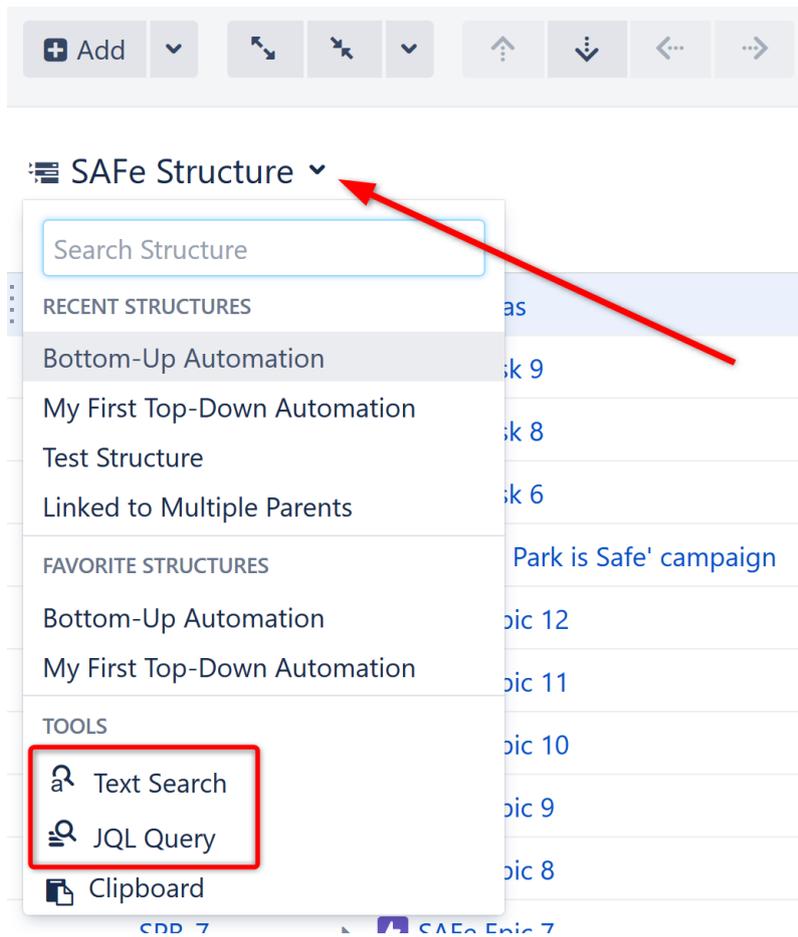
**i** Apart from choosing structures, you can also use this drop-down to run text and JQL searches, see the clipboard contents and - if you have Structure.Pages installed - search for Confluence pages. All of these options are available under the Tools section of the menu, when using [Structure Board](#)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

## 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.

The screenshot shows a Jira search results table for the query 'Theme'. The table has columns for Key, Summary, Story Points, Time Spent, Progress, TP, Assignee, Status, and WSJF (Basic). The results are as follows:

Key	Summary	Σ Story Points	Σ Time Spent	Progress	TP	Assignee	Status	WSJF (Basic)
SPF-4	SAFe Theme 4					Bob	BACKLOG	800
SPF-3	SAFe Theme 3					M. Reynolds	IN PROGRESS	400
SPF-2	SAFe Theme 2					M. Reynolds	IN PROGRESS	1600
SPF-1	SAFe Theme 1					M. Reynolds	SELECTED FOR	800
SP-9	Build a transparent d	9	1d			C. Bacca	IN PROGRESS	282
MKT-4	'Theme Park is Safe' c					Demo User	TO DO	800
MKT-1	Anti-PR campaign to		15w			Man in Black	IN PROGRESS	196
META-10	Theme Park Construc					Bob	OPEN	400

Showing 8 items

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](#) as necessary.

The **structure panel toolbar** also works for search results the same way it works for structures. You can apply [sorting](#), additional [filtering](#) and more complex [transformations](#).



Search only looks for issues from [structure-enabled projects](#).

### 2.1.5 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](#)
- [Structure on the Issue Page](#)
- [Structure on the Project Page](#)
- [Structure on Agile Boards](#)
- [Structure Gadget](#)

## 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.

**Structure** ⚙️

☰ Manually Built Structure 🏠 ... ||| ▾

Key	Summary	Progress	TP	Assignee	
	▾ 📁 Theme Park Construction	<div style="width: 50%;"><div style="background-color: green; height: 10px;"></div></div>			
TP-124	▾ 🗨️ Site preparations	<div style="width: 75%;"><div style="background-color: green; height: 10px;"></div></div>	🗨️ ⬆️	Bob	
📌 ● SP-9	▾ 🏗️ Build a transparent dome	<div style="width: 60%;"><div style="background-color: green; height: 10px;"></div></div>	🏗️ ⬆️	Bob	⚙️ ▾
QA-7	📄 Check seismic activity	<div style="width: 20%;"><div style="background-color: green; height: 10px;"></div></div>	📄 ⬆️	Unassigned	

Showing 4 items 🔗 Open ⓘ Info

The structure will open on Structure Board with the same [filters](#) and [transformations](#) 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.

## 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](#) gives you access to the main functions for building and working with structures.
- **Working Panels.** The left panel always displays the [structure widget](#) or [search results](#), while the right panel can display another structure widget, [issue details](#), [history](#) 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.

The screenshot displays the Jira Software interface. The top navigation bar includes 'Jira Software', 'Dashboards', 'Projects', 'Issues', 'Boards', 'Structure', and '+ Create'. The main content area is divided into two panels. The left panel, titled 'SAFe Overview', shows a hierarchical tree of items with columns for 'Summary', 'WSJF (Basic)', and 'Progress'. The right panel, titled 'Issue Details', shows the details for a 'Sub-task 1' issue, including its type, status (IN PROGRESS), priority (Major), resolution (Unresolved), and assignee (H. Solo).

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](#). From the Manage Structure page, you can browse and search for available structures.
- Open another structure and [switch between structures](#) 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`

### **i** 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*)

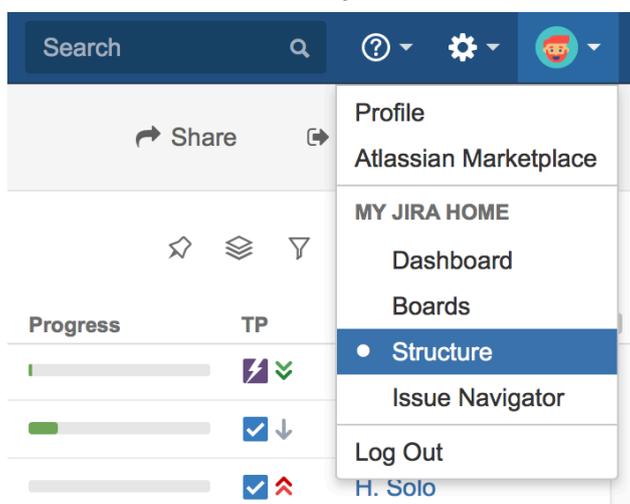
### **✓** Save Time!

You can make the Structure Board your [Jira Home page](#).

## Making Structure Board Your Jira Home

If you want to go straight to the [Structure Board](#) 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.

## Structure on the Issue Page

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

 A screenshot of the Jira issue page for 'Relocate trees' (SP-5). The page shows the issue details, including a 'Sub-Tasks' section with a 'Structure' widget. The Structure widget displays a table of sub-tasks with columns for 'Key', 'Summary', 'Progress', and 'TP'. The 'Relocate trees' sub-task is highlighted. To the right of the Structure widget is a 'Time Tracking' section with a 'Time Tracking' header and a 'Time Tracking' sub-section. The Time Tracking section shows 'Include structure sub-issues' checked, and provides summary statistics for 'Estimated', 'Remaining', and 'Logged' time. Red arrows point to various UI elements: 'Selected Structure' points to the Structure widget title, 'Structure tools' points to the 'Pin' icon, 'Views menu' points to the 'More' icon, 'Options' points to the gear icon, and 'Adjusted time tracking' points to the Time Tracking section.
 

Key	Summary	Progress	TP
TP-124	Site preparations	<div style="width: 100%;"></div>	
SP-10	Move stuff to another	<div style="width: 100%;"></div>	
SP-5	Relocate trees	<div style="width: 100%;"></div>	
SP-13	Find the	<div style="width: 100%;"></div>	
SP-12	Find ne	<div style="width: 100%;"></div>	
SP-16	Rent th	<div style="width: 100%;"></div>	

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

## Pinned Issue

The issue itself is automatically located and [Pinned](#) 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](#) 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](#) 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

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](#).) 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](#)).

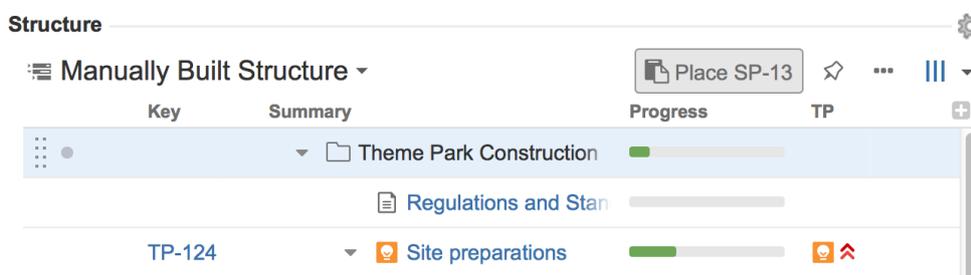
To switch to a different structure, simply click the name of the currently displayed structure and select the one you want to see. You will see those structures that contain this issue in the top section of the displayed menu.

 Structures where this issue is added through [Automation](#) 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](#) and select another view for the displayed structure.
- Click Options icon to open [Structure Options for the Issue Page](#).

 An asterisk appears next to the view name if it has been locally [adjusted](#).

## 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.

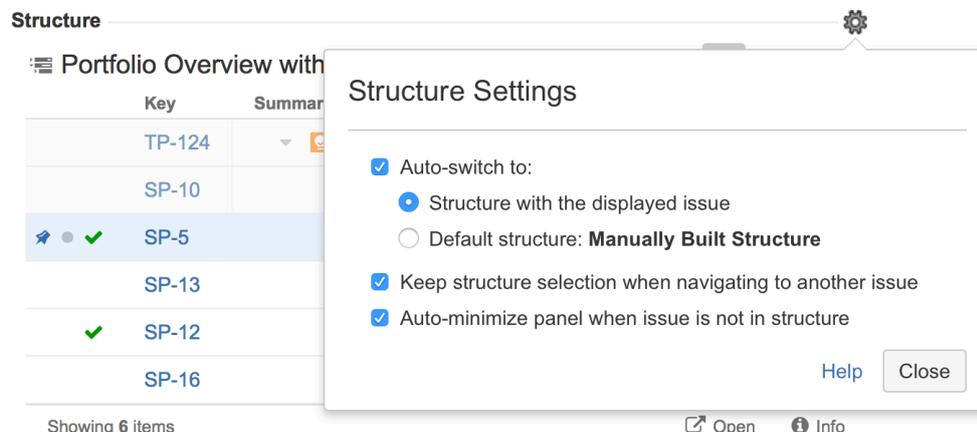
If the time tracking section is not present, it means that neither the current issue nor its sub-issues have any time tracking info.

## 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](#) 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:

<b>Auto-switch</b>	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 <i>current</i> structure).
<b>Auto-switch: structure with displayed issue</b>	When this auto-switch mode is selected, Structure looks for a structure that contains the issue displayed on the page.
<b>Auto-switch: default structure</b>	When this auto-switch mode is selected, the <b>Default Structure</b> for the issue's project will always be selected (even if the issue is not in that structure).

<p><b>Keep structure when navigating</b></p>	<p>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).</p> <div data-bbox="395 450 1423 660" style="border: 1px solid #c8e6c9; padding: 10px;"> <p> We recommend leaving this option on. This will prevent you from unintentionally switching structures while reviewing a structure's issues.</p> </div>
--	--

 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.

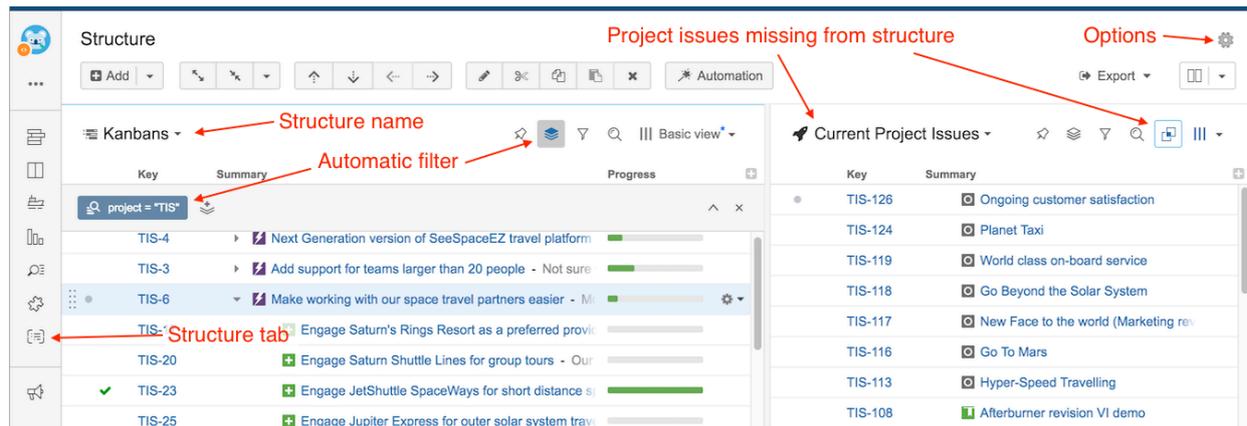
## Options Scope and Default Options

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](#) page.

## Structure on the Project Page

If a project is [enabled for Structure](#), 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](#), 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 Double Grid layout is selected automatically.

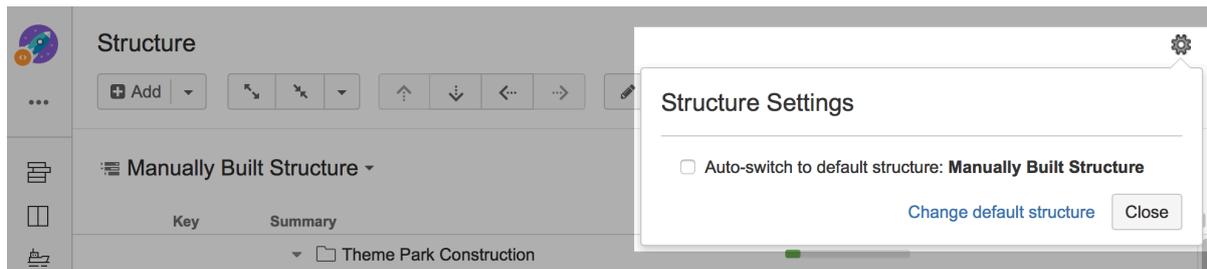
The primary (left) panel displays the most recently viewed structure or the default structure for the current project (as defined in the Options dialog). You can quickly switch to another structure by clicking the structure name and selecting the desired structure.

The secondary (right) panel shows issues from the current project that are not part of the selected structure. This allows to quickly place other issues from the project into the 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.

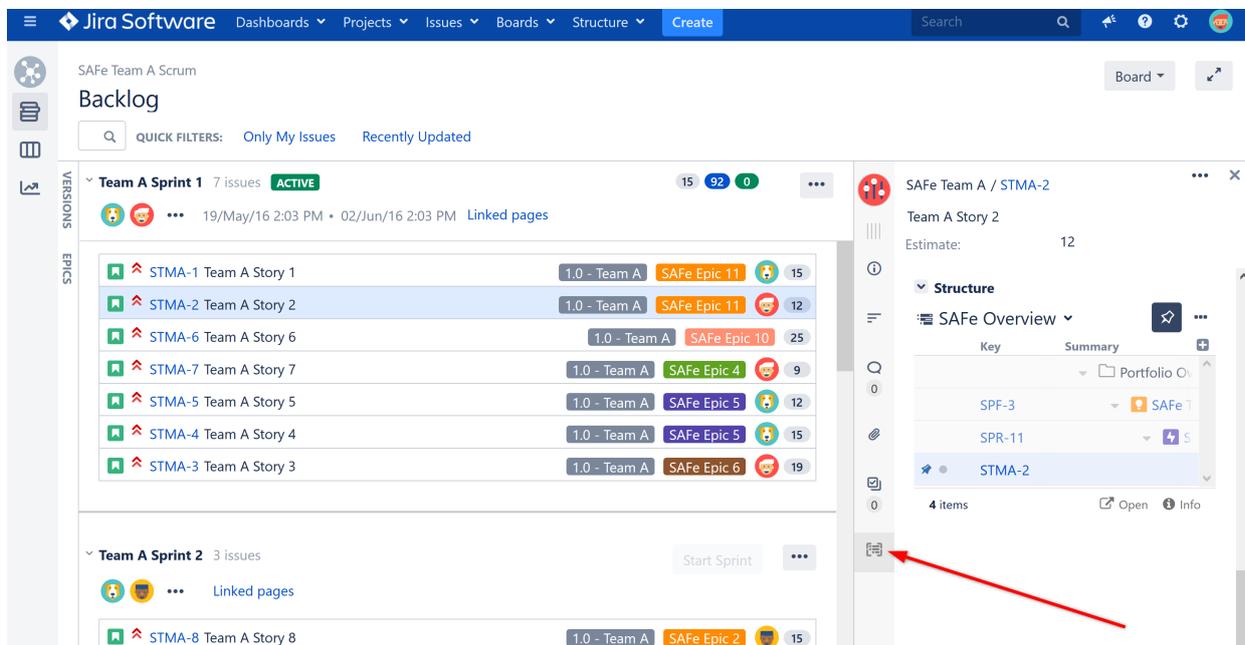
**i** The default value for this option can be configured by the JIRA Administrator on the [Structure Defaults](#) page.

## Perspectives are Unavailable

It is not possible to [share a perspective](#) from the project page.

## 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](#) (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](#) 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.

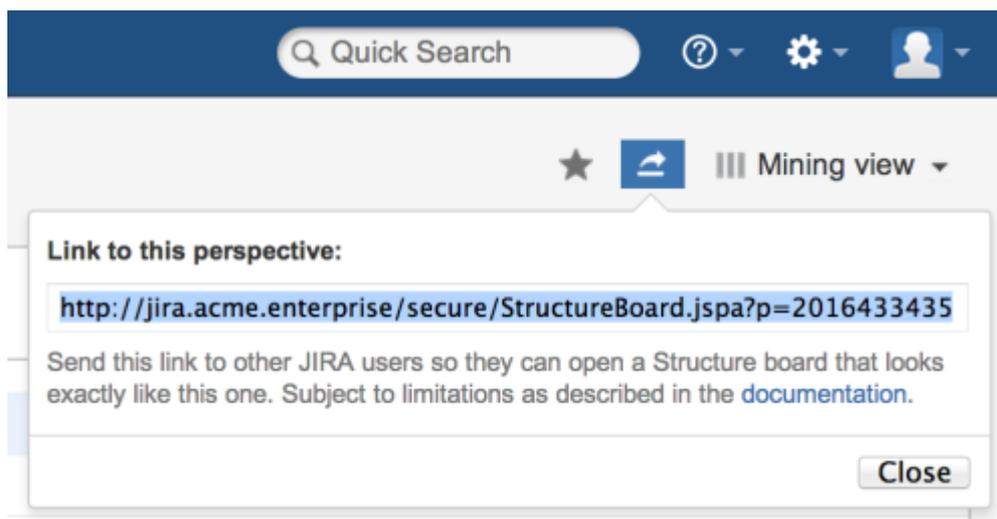
 Only the users who have [access to Structure](#) will see the Structure tab on Agile boards.

### 2.1.6 Sharing a Perspective

Structure Perspectives is a feature that lets you store the way you see a structure in the form of a permanent link which can be published or sent to another person.

To create a perspective:

1. Open [Structure Board](#)
2. Click the "Share perspective" button. A message with the link will appear:



3. Copy the link and save it, publish it, or send it to persons you want to share it with.

To use a perspective:

1. Follow the link you've received. This will open a Structure Board in Perspective mode, that is, the Structure Board will look mostly the way it looked when the perspective was created.

 A special **Perspective View** 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.

When creating a perspective, please consider the following:

1. If you send the link to someone who has no access to the Structure Plugin in general, or to the individual structure for which the perspective was created, they won't be able to use your link.
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. A structure opened in Perspective mode will always show the current issue hierarchy (it will contain all the changes that were made after the perspective was created).

 You can open structure history and select the latest change before creating a perspective. This way, users will always see structure in history mode when opening your 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](#).
5. Once created, a perspective becomes accessible to any person who has access to the structure for which that perspective was created.

## 2.2 Basic Operations

### 2.2.1 Adding Items to a Structure

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

- [Issues](#)
- [Folders](#)
- [Memo](#)
- [Generators \(Automation\)](#)

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

### Adding Issues

Issues can be added to a structure from the [Structure Board](#) or their own [Issue Page](#).

### Adding issues from Structure Board

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

After selecting the issue(s) you need, add them to your structure using [drag-and-drop](#) or [copy /paste](#).

The screenshot shows two panels side-by-side. The left panel, 'John's Structure', has a table with columns: Key, Summary, Progress, TP, Assignee. It lists several issues like 'Performance', 'Fuel Efficiency', 'Strange ef', 'Double the G p', 'Warp drive mod', and 'Automatic hyper-driv'. The right panel, 'JQL', has a table with columns: Key, Summary. It lists issues like 'Flatten building site', 'Triple Helix Rollercoaster', 'Overunity Ferris Wheel', 'Theme Park is Safe' campaign, 'Large Hadron Merry-go-Round', 'Journey to the center of the Earth', 'Defense capabilities of Hyperdrive Engin', 'Performance', 'Hyperdrive', and 'Galactic Empire'. A red circle highlights a three-dot menu icon in the right panel, and a red arrow points from it to the 'OFW-1' issue in the left panel.



You can also add issues from another structure. Simply [open the structure in the secondary panel](#), 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.

The screenshot shows a 'Structure' panel with a table of issues. The table has columns: Key, Summary, Σ Story P, Σ Time Sp, Progress, TP, Assignee, Status, and WSJF (Basic). The table lists several issues, including 'Regulations and', 'TP-124 Site preparation', and 'TP-31 Marketing + PR'. A red arrow points from a 'Paste' icon in the top right corner of the table to the 'Regulations and' issue row.



Issues can be added and removed automatically using [Automation](#).

## Folders

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

Fun with Folders ▾

☆ ⚙️ 🔍 ||| Basic view ▾

Key	Summary	Σ Story Points	Σ Time Spent	Progress	TP	Assignee	Status	WSJF (Basic)
	Project A	46	2d	<div style="width: 100%; height: 10px; background-color: green;"></div>				
STMA-1	Team A Story 1	15	1d	<div style="width: 75%; height: 10px; background-color: green;"></div>	🟢 ⚠️	Anna M.	IN PROGRESS	317
STMA-36	Task			<div style="width: 0%; height: 10px; background-color: green;"></div>	🟡 ⚠️	Unassigned	TO DO	800
STMA-2	Team A Story 2	12	6h	<div style="width: 50%; height: 10px; background-color: green;"></div>	🟢 ⚠️	Anna M.	IN PROGRESS	309
STMA-3	Team A Story 3	19	2h	<div style="width: 25%; height: 10px; background-color: green;"></div>	🟢 ⚠️	Anna M.	IN PROGRESS	286
	Project B	23	2h	<div style="width: 10%; height: 10px; background-color: green;"></div>				
STMB-1	Team B Story 1	19	2h	<div style="width: 50%; height: 10px; background-color: green;"></div>	🟢 ⚠️	Mary (Inactive)	TO DO	306
STMB-43	Bug 1		2h	<div style="width: 100%; height: 10px; background-color: green;"></div>	🟢 ⚠️	Unassigned	DONE	
STMA-35	Bug 2			<div style="width: 0%; height: 10px; background-color: green;"></div>	🟡 ⚠️	Unassigned	TO DO	800
STMB-2	Team B Story 2	4		<div style="width: 0%; height: 10px; background-color: green;"></div>	🟢 ⚠️	Mary (Inactive)	IN PROGRESS	342

Some common uses for folders include:

- Organizing issues into specific categories
- Separating different projects or different parts of a project
- Creating different [Automation](#) 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](#))
- Placing a structure within a structure (it's not necessary to use folders, but we recommend it)

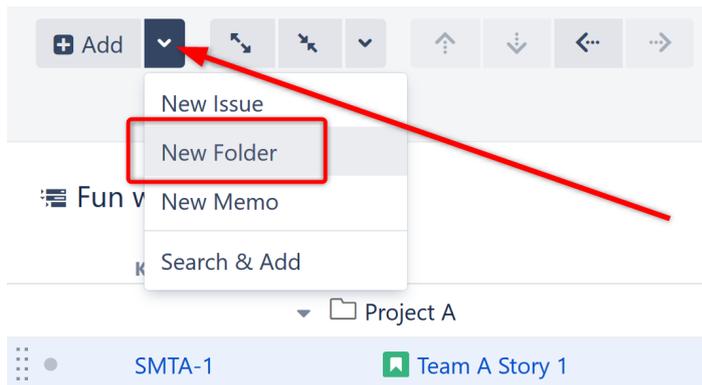


[Group generators](#) make their own folders to group items by a common attribute.

## 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](#) or higher to add folders to a structure.

## Memo

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

Simple Structure ▾ ⚡ ☆ ⌘ 🔍 ||| Basic view\* ▾

Key	Summary	Status	Progress
STMB-31	Story 1	IN PROGRESS	<div style="width: 75%;"></div>
STMB-48	Task 1.1	TO DO	<div style="width: 0%;"></div>
	Memo - Add additional information anywhere you need it!		
	Flag - You can change a memo's icon or color.		
STMB-49	Task 1.2	TO DO	<div style="width: 0%;"></div>
	Parent Memo - Memos are part of the hierarchy. You can place issues be		<div style="width: 75%;"></div>
STMB-37	Story 4	IN PROGRESS	<div style="width: 75%;"></div>
STMA-36	Task	TO DO	<div style="width: 0%;"></div>

Structure with memos

Memos can serve a variety of purposes within a structure:

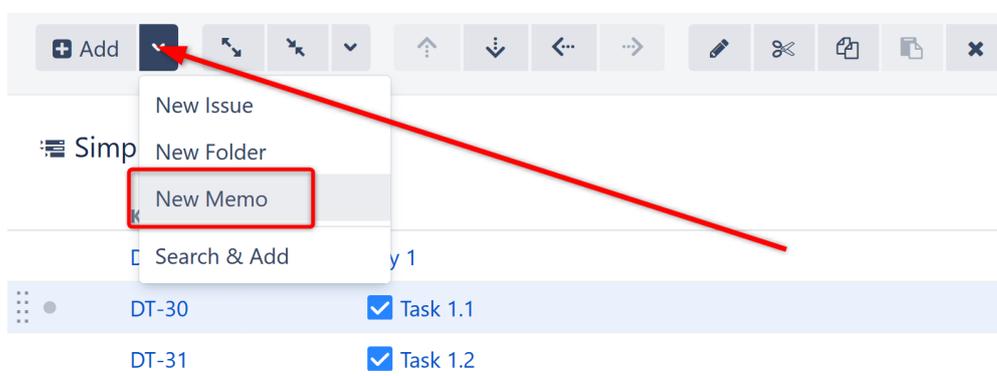
- Add notes or reminders that pertain to the structure or project as a whole, rather than just a single issue (for that, try a [Notes column](#))
- Add high-level requirements directly to your structure
- Add a placeholder for other items

- 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

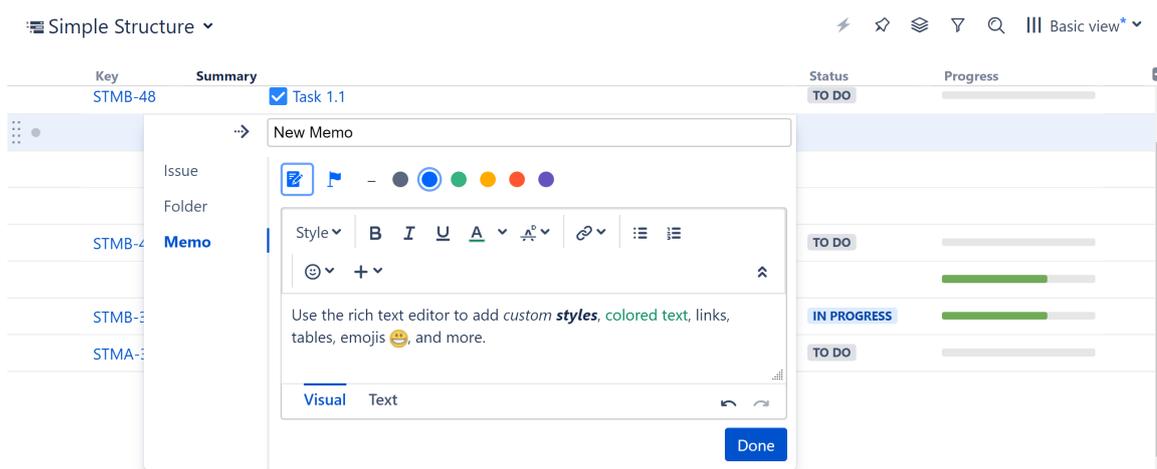
To add a memo to a structure:

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**.



Add New Memo

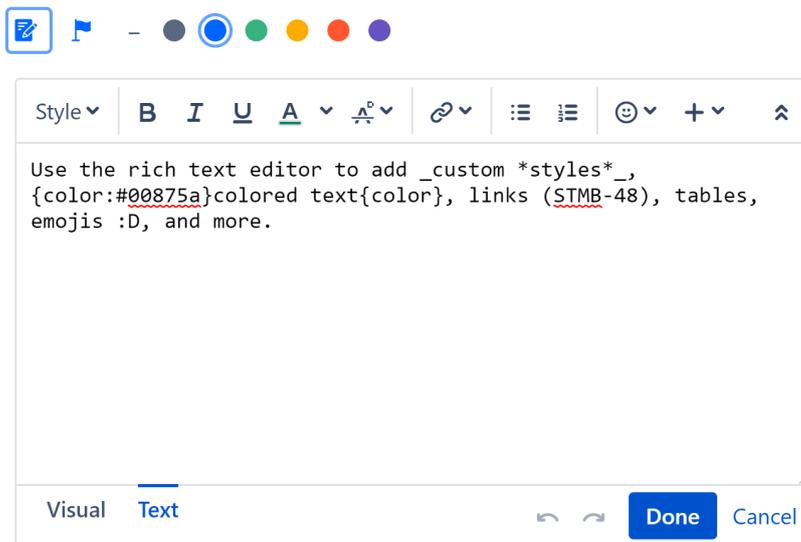
4. Choose the icon and color. Add text and any necessary markup using the built-in rich text editor.



Add new memo

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](#) for a complete list of available formatting options and conventions.

## New Memo



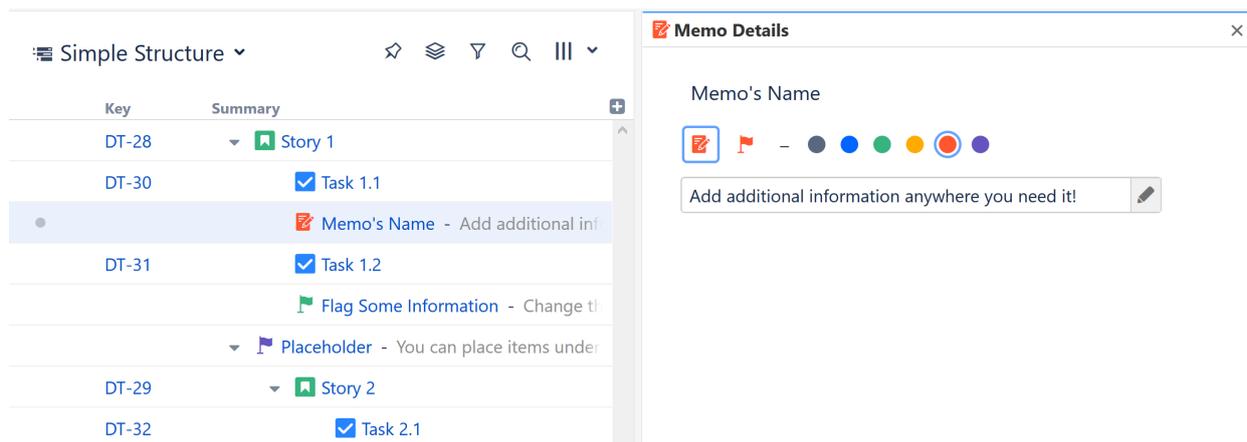
## Text mode



**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.



### Editing a memo

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.

## 2.2.2 Moving Items within Structure

### Basic Moves

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.

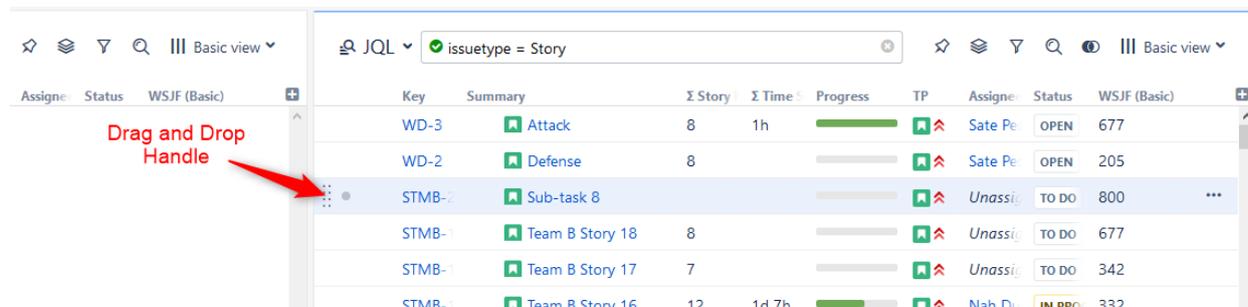
### Advanced Moves

To move items more than one space at a time, use Drag-and-Drop or Cut & Paste.

## Drag-and-Drop

Using the mouse, you can [Drag-and-Drop](#) items anywhere in the structure, even moving them from beneath one parent item to another.

To move an item with the mouse, use the Drag and Drop Handle on the far left of the item's row.



## Cut & Paste

[Cut & Paste](#) also allows you to move items to any location within the structure. It can even be used to copy the hierarchy from one structure to another.



Moving items with Drag-and-Drop or Cut & Paste can be [undone](#).

## Moving Multiple Items

You can select multiple items and move them all together, using any of the methods mentioned above. Just keep in mind the following rules:

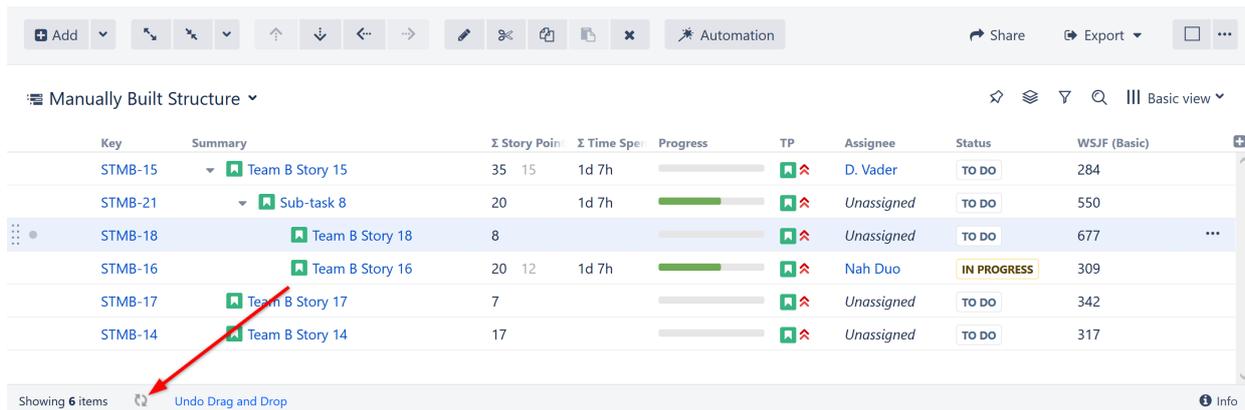
- Moving items with the toolbar or keyboard (Move Up/Down and Level Up/Down) only works if all of the selected items are at the same level in the hierarchy and under the same parent.
- Drag-and-Drop and Copy & Paste support multiple-item moves in any configuration.

See also: [Selecting Multiple Items](#)

## Back-to-Back Moves

When you make changes in a structure, they are uploaded to the server asynchronously, allowing you to continue working, regardless of any network delay. If you need to make several moves, one after another, you can do so without having to wait for each change to take effect on the server side.

If changes are still being processed, the **synchronization** icon will appear in the status bar.



Key	Summary	Σ Story Points	Σ Time Spent	Progress	TP	Assignee	Status	WSJF (Basic)
STMB-15	Team B Story 15	35 15	1d 7h	<div style="width: 100%;"></div>	🟢 🔴	D. Vader	TO DO	284
STMB-21	Sub-task 8	20	1d 7h	<div style="width: 100%;"></div>	🟢 🔴	Unassigned	TO DO	550
STMB-18	Team B Story 18	8		<div style="width: 100%;"></div>	🟢 🔴	Unassigned	TO DO	677
STMB-16	Team B Story 16	20 12	1d 7h	<div style="width: 100%;"></div>	🟢 🔴	Nah Duo	IN PROGRESS	309
STMB-17	Team B Story 17	7		<div style="width: 100%;"></div>	🟢 🔴	Unassigned	TO DO	342
STMB-14	Team B Story 14	17		<div style="width: 100%;"></div>	🟢 🔴	Unassigned	TO DO	317

Showing 6 items Undo Drag and Drop

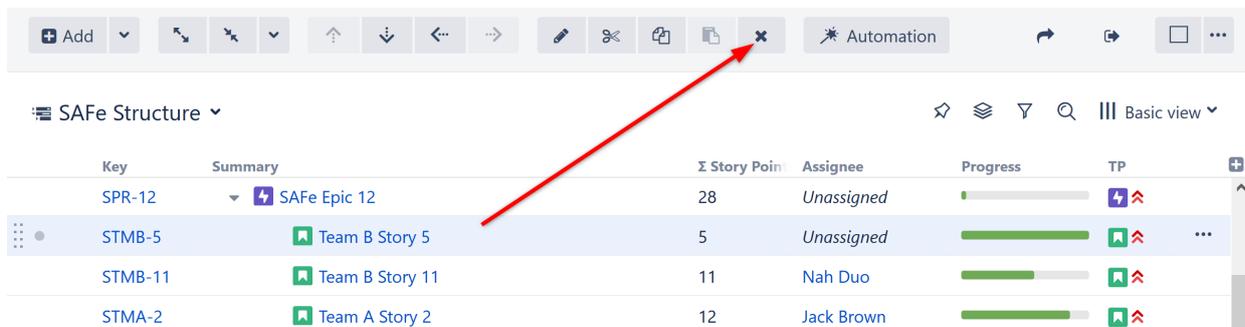
## Moving Items with Automation

If you used [Automation](#) to build all or part of a structure, the content it adds cannot be moved as freely as content which has been manually added to a structure. 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.

In order to move items freely within a generated structure, you may need to enable [Manual Adjustments](#).

### 2.2.3 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.



Key	Summary	Σ Story Points	Assignee	Progress	TP
SPR-12	SAFe Epic 12	28	Unassigned	<div style="width: 100%;"></div>	🟢 🔴
STMB-5	Team B Story 5	5	Unassigned	<div style="width: 100%;"></div>	🟢 🔴
STMB-11	Team B Story 11	11	Nah Duo	<div style="width: 100%;"></div>	🟢 🔴
STMA-2	Team A Story 2	12	Jack Brown	<div style="width: 100%;"></div>	🟢 🔴

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](#) and click **Delete**.

🟢 Removing items can be [undone](#).

## 2.2.4 Undoing Changes

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

- [Adding](#) items from search results;
- [Removing](#) items from a structure;
- [Drag-and-Drop](#);
- The Paste operation of a [Cut & Paste](#) sequence.

When you perform an operation that can be undone, a corresponding hyperlink appears in the footer at the bottom of the Structure widget. 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](#), 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 the new functionality for Undo and it will be added in the future versions.



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](#).

## 2.2.5 Working with Issues

Structure allows you to view and edit issues without leaving the Structure widget. Select one of the following articles to see how you can work with issues from within a structure.

### 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:

Click the link...

Or switch to the Grid + Details layout

Key	Summary	Σ Story Points	Assignee	Progress
SPR-11	SAFE Epic 11		M. Reynolds	
SPR-5	SAFE Epic 5	42	M. Reynolds	
STMA-4	Team A Story 4	15	C. Bacca	
STMA-5	Team A Story 5	12	C. Bacca	
STMA-16	Team A Story 16	15	Unassigned	
SPR-4	SAFE Epic 4	22	M. Reynolds	
STMA-7	Team A Story 7	9	Jack Brown	
STMA-14	Team A Story 14	13	Unassigned	



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.

## 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](#), [view and add comments](#), [share](#), [view history](#), [view development information](#), and more. For specific information on working with and editing issues, please refer to the [Jira documentation](#).

Issue Details

SAFE Team B / STMB-16

Team B Story 16

Edit Comment Assign More To Do In Progress Done Admin Share Export

Details

Type: Story

Status: **IN PROGRESS** (View Workflow)

Priority: Major

Resolution: Unresolved

People

Assignee: Nah Duo

Assign to me

To see details for another issue in the structure, simply [select another issue](#) 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.

## 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.

**i** You can adjust the default views in [View Settings](#).

## 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.

The screenshot shows the Structure Board interface. The main panel displays a list of items with columns for Key, Summary, Σ Sto, Assi, Progre, and TP. The Issue Details panel is open on the right, showing details for 'Team A Story 4'. A red arrow points to the vertical border between the two panels, indicating it is draggable.

Key	Summary	Σ Sto	Assi	Progre	TP
SPF	SAFe Epic 11		M. F.		
SPF	SAFe Epic 5	42	M. F.		
STI	Team A Story 4	15	C. B.		
STI	Team A Story 5	12	C. B.		
STI	Team A Story 16	15	Unc.		
SPF	SAFe Epic 4	22	M. F.		
STI	Team A Story 7	9	Jack		
STI	Team A Story 14	13	Unc.		

The Issue Details panel for 'Team A Story 4' shows the following information:

- Type: Story
- Status: TO DO (View Workflow)
- Priority: Major
- Assignee: C. Bacca
- Reporter: M. Reynolds

**i** 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.

## Details and Secondary Panels

If you have the [secondary panel](#) 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.

## 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 **o** or **o** (also **PgUp**, **PgDn**, **Home**, **End**, or **,**, **,**, **,**) scroll the details panel, while all other [Structure shortcuts](#) 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.

## 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.

### 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.

Manually Built Structure

Key	Summary	Σ Story P
STMB-21	Sub-task 8	20
STMB-16	Team B Story 16	12

Issue

Folder

Categories  Copy from *STMB-16 Team B Story 16*

Project\* SAFe Team B (STMB)

Issue Type\* Story

Switch to Dialog

Done

## ✔ 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**

**i** 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](#).



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.

## 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.

Issue

Folder

Categories  Copy from *STMB-21 Sub-task 8*

Project\* SAFe Team B (STMB)

Issue Type\* Story

[Switch to Dialog](#) Done

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.

Create Issue & Add to Structure Configure Fields

Project\* Agile Project  
Only projects enabled for Structure are offered.

Issue Type\* Bug

Summary\*

Priority Major

Due Date

Component/s **None**

Affects Version/s **None**

Fix Version/s **None**

Assignee Automatic  
[Assign to me](#)

Reporter\* admin  
Start typing to get a list of possible matches.

Environment

Switch to Panel Don't Add to Structure  Create another Create Cancel

For example operating system, software platform and/or hardware specifications (include as appropriate for the

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.

## 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.

## 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**:

<b>Enter</b> or <b>Tab</b>	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.
<b>Ctrl+Enter</b> / <b>Cmd+Enter</b>	Toggles cloning mode ( <b>Categories: Copy</b> checkbox).
<b>Alt+Enter</b> / <b>Option+Enter</b>	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.

## 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.

## Editing Issues

You can quickly edit issues from within Structure, without leaving the current page. 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**

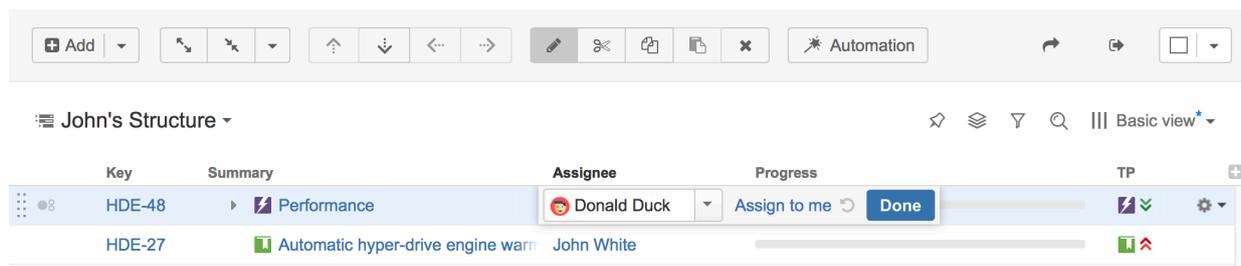
- ✔ If the value is a link (like in the Summary or Assignee fields), you can still double-click it – the browser will not open the link and will start editing instead.

## 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](#)).

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



Key	Summary	Assignee	Progress	TP
HDE-48	Performance	Donald Duck	Assign to me <span>Done</span>	
HDE-27	Automatic hyper-drive engine warn	John White		

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**.

## Learn More

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



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](#) will be displayed at the far left of the item row.

## 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.

TP	Assignee	Status	WSJF (Basic)
	Unassigned	BACKLOG	489
	<input type="text" value="Jack Brown"/>	Done	718
	Nan Duo	IN PROGRESS	349
	Jack Brown	IN PROGRESS	363

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, or
- Switch to editing another issue.

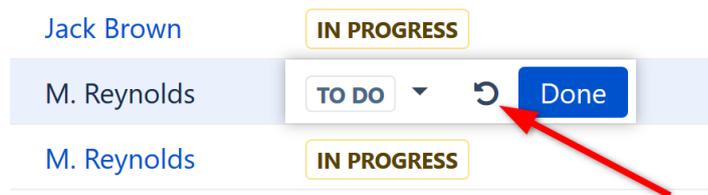


If your Jira is configured to send e-mail notifications about changes, then a notification will be sent as soon as you have finished editing an issue - see [On E-mail Notifications](#)

## 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!).

TP	Assignee	Status	WSJF (Basic)
	C. Bacca	BACKLOG	489
	Jack Brown	IN PROGRESS	718
	M. Reynolds	Assign to me	
	Jack Brown	IN PROGRESS	363

## 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)

## Keyboard Shortcuts in Edit Mode

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

### Entering Edit Mode

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

### Keyboard Shortcuts in the Edit Mode

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

Keyboard Shortcut	Action
	Edit previous editable field. If the current field is the first editable field for the selected issue, start editing previous issue.
<b>Ctrl+Alt+</b>	Edit the same field of the next editable issue.
<b>Ctrl+Alt+</b>	Edit the same field of the previous editable issue.
<b>Ctrl+Alt+</b>	Edit next editable field. Unlike <b>Tab</b> , this combination will not move editing to the next issue.
<b>Ctrl+Alt+</b>	Edit previous editable field. Unlike <b>Shift+Tab</b> , this combination will not move editing to the previous issue.
<i>or Alt+</i> <i>(in drop-downs)</i>	Opens the drop-down list or selects the next value in the list. If the drop-down is shown, use <b>Enter</b> to select a value or <b>Escape</b> to cancel selection.
<b>Alt+</b> <i>(in date/time fields)</i>	Opens the date picker. Use arrows to navigate dates in the date picker; use <b>Enter</b> to select a date or <b>Escape</b> 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 <i>(for checkboxes and radio buttons)</i>	Move between multiple fields on the same editor (for example, between the checkboxes of a Multiple Checkboxes custom field).
<b>Space</b>	Select / unselect a checkbox or a radio button.
, , <b>Shift+</b> , <b>Shift+</b>	Select / unselect values in a Multi-Select custom field.

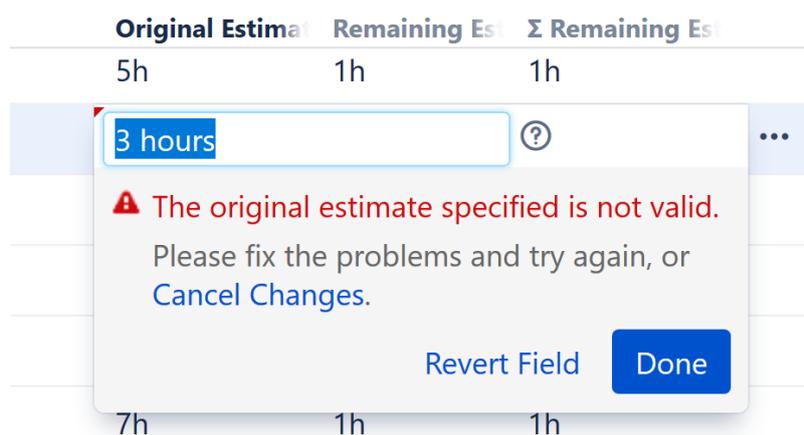


Note that the **Tab** key moves editing to the next cell, so if you have multiple input fields on a single field editor, you need to use arrow keys to switch between them.

See Also: [Keyboard Shortcuts](#)

## 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.

## 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.

The following Jira fields are editable from Structure Gadget:

- Summary
- Assignee
- Issue Type
- Priority
- Reporter
- Security Level
- Original Estimate
- Remaining Estimate

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.

[Structure Notes](#) can also be edited in Structure Gadget.

## On E-mail Notifications

Usually, when an issue is edited, an e-mail notification is sent to everyone involved with that issue.

When editing an issue within Structure, the changes are saved on the server and the e-mail notification is sent when you:

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

If you switch from editing one field of an issue to editing another field of the same issue immediately, your changes will only be sent to the server once you hit Done or Enter, or switch to a new issue – meaning just one email will be sent for all changes to that issue.

On the other hand, if you edit a field, click Done, then edit another field - that's two edits, so there will be two notifications.



If you need to change several fields of an issue and avoid multiple e-mails being sent, edit one field and then navigate to the next field. Only hit **Done** or **Enter** when you have finished editing all fields.

## Bulk Change

With Structure, you can quickly [select multiple issues](#) and edit them using the Jira bulk operation wizard.

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](#) 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.

## Cloning Multiple Issues

Structure allows you to copy the whole structure and clone all issues in the structure. See [Copying Structure and Cloning Issues](#).

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](#).
2. Use **Copy** action on the toolbar (or hit Ctrl+C / Command+C) to copy the issues to the [Issue Clipboard](#).
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](#). 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**.

## Using Jira Actions

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

## 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.

Key	Summary	TP	Due Date	Fix Version/s	Assignee	Original Estim	Remaining Es	Σ Remaining E	
✓ STR-3	🔗 Formulas	🔗 ⬆️			Unassigned	18w 2d 5h			
• ✓ STMB-23	📄 Sub-task 9	🔗 ⬆️		2.0 - Team B	Mary (Inactive)	5h	4h	4h	⋮
✓ STMB-22	📄 Sub-task 8	🔗 ⬆️		2.0 - Team B	Mary (Inactive)	4h	2h		
✓ STMB-19	📄 Sub-task 6	🔗 ⬆️		1.0 - Team B	Nah Duo	5h	1h		
MKT-4	▶️ 🔗 'Theme Park is Safe' c	🔗 ⬆️		1	Demo User				
SPR-12	▶️ 🔗 SAFe Epic 12	🔗 ⬆️			C. Bacca				



The Jira Actions menu is unavailable when the [Issue Details panel](#) is opened.

## 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](#), keyboard shortcuts are always applied to the viewed issue - regardless of the selection in the structure.

## 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](#), most actions do not require page a reload.

## 2.3 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 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.

### 2.3.1 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.

### 2.3.2 Types of Generators

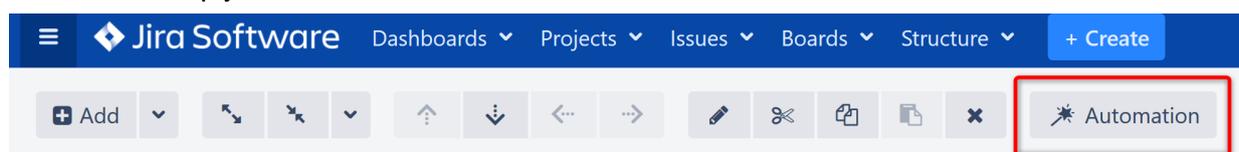
### 2.3.3 Generator Scope

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.

### 2.3.4 Adding a Generator

#### 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.

#### Adding a Generator

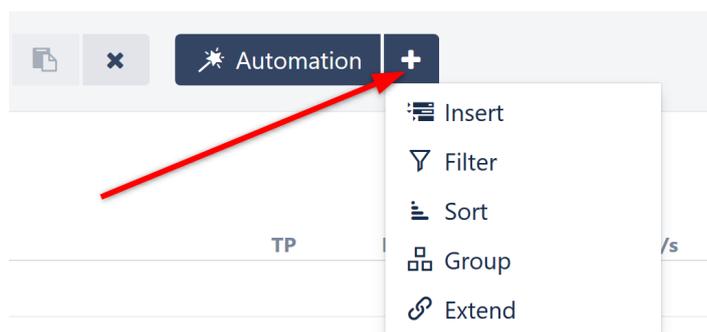
Before adding a generator, you must select where you want to place it within the structure. A generator's placement defines its scope:

- 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.

**i** You can only add generators under static parts of the 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](#).

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

Key	Summary	Progress
	Manually Built Structure	<div style="width: 10%;"></div>
	Sorted by Progress	<div style="width: 10%;"></div>
	<ul style="list-style-type: none"> <li>Theme Park Construction</li> </ul>	<div style="width: 20%;"></div>
TP-124	<ul style="list-style-type: none"> <li>Site preparations</li> </ul>	<div style="width: 40%;"></div>
SP-15	Install entrance checkpoints	<div style="width: 60%;"></div>
SP-3	Remove waste rock and soil	<div style="width: 80%;"></div>

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

### 2.3.5 Types of Generators

There are several types of generators available by default:

## Insert Generators

Insert generators allow you to automatically add issues to the structure. 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](#) 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](#) or [Sort Generator](#))
- 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](#).

## Types of Inserters

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

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.

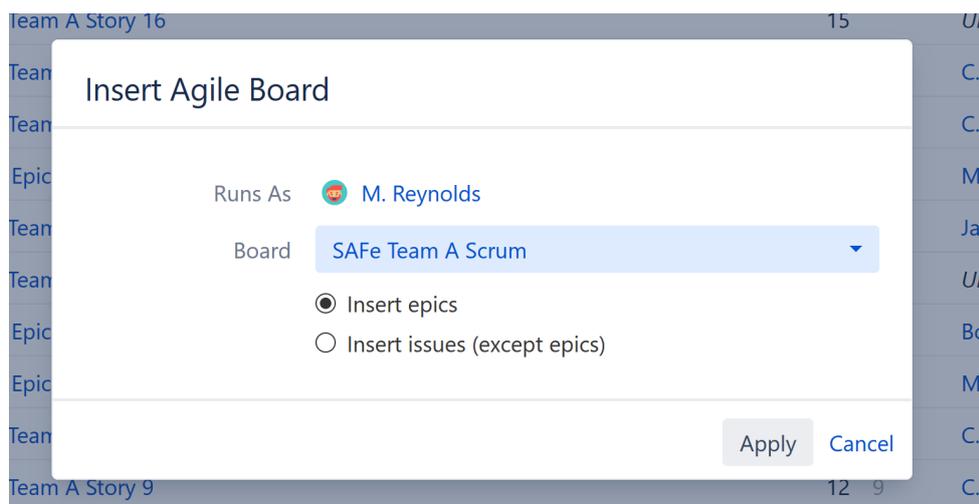
## Always Up-To-Date

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.

## Agile Board Inserter

The Agile Board Inserter 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](#) or [Group](#) 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.

Key	Summary	Σ Story Points	Assignee	Pr.	TP
Agile Board Inserter					
	+ Insert epics from "SAFe Program Kanban"				
SPR-2	SAFe Epic 2		M. Reynolds		
SPR-1	SAFe Epic 1		M. Reynolds		
SPR-4	SAFe Epic 4		M. Reynolds		
SPR-6	SAFe Epic 6		Bob		
SPR-11	SAFe Epic 11		M. Reynolds		



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](#), 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](#) 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](#) or [Filter](#) generators to tailor the results to your specific needs – or use [Transformations](#) to temporarily sort or filter your results.

✔ You can expand and customize Inserter results using the [Extend](#), [Filter](#), [Sort](#) and/or [Group](#) generators.

## JQL Query Inserter

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

The screenshot shows a dialog box titled "Insert JQL Query". It has the following fields and controls:

- Runs As:** Admin (with a user icon)
- JQL Query:** project = "SAFE Team A" AND status = "In Progress" (with a green checkmark icon)
- Limit:** 1000
- Text below Limit:** Maximum number of issues to insert (leave empty for no limit)
- Buttons:** Apply, Cancel

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

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.

JQL Query Inserter ▾

☆ ⚙️ 🔍 Basic view ▾

Key	Summary	Progress	Status	Assignee	Icons
JQL Query Inserter		<div style="width: 50%; background-color: green; height: 10px;"></div>			
+ Insert issues: project = "SAFe Team A"					
STMA-18	Sub-task 2	<div style="width: 20%; background-color: green; height: 10px;"></div>	IN PROGRESS	Albert	🔗 ⬆️
STMA-17	Sub-task 1	<div style="width: 20%; background-color: green; height: 10px;"></div>	IN PROGRESS	H. Solo	🔗 ⬆️
STMA-7	Team A Story 7	<div style="width: 60%; background-color: green; height: 10px;"></div>	IN PROGRESS	Jack Brown	🔗 ⬆️
STMA-6	Team A Story 6	<div style="width: 40%; background-color: green; height: 10px;"></div>	IN PROGRESS	Claire T.	🔗 ⬆️
STMA-5	Team A Story 5	<div style="width: 50%; background-color: green; height: 10px;"></div>	IN PROGRESS	Anna M.	🔗 ⬆️
STMA-3	Team A Story 3	<div style="width: 10%; background-color: green; height: 10px;"></div>	IN PROGRESS	Jack Brown	🔗 ⬆️
STMA-2	Team A Story 2	<div style="width: 60%; background-color: green; height: 10px;"></div>	IN PROGRESS	Jack Brown	🔗 ⬆️
STMA-1	Team A Story 1	<div style="width: 40%; background-color: green; height: 10px;"></div>	IN PROGRESS	Anna M.	🔗 ⬆️

## 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:

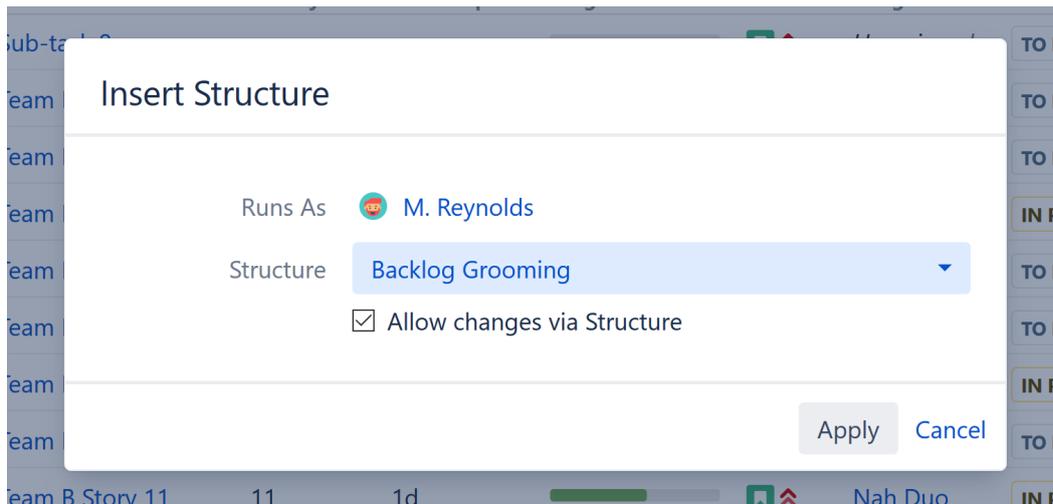
- You need to include issues from multiple projects (or even every structure-enabled project you have access to)
- You need to add issues based on a specific Jira field. For example, you may want to add every issue assigned to a specific team, add every bug, only include issues from certain versions, 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](#), [Filter](#), [Sort](#) and [Group](#) generators.

## 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.

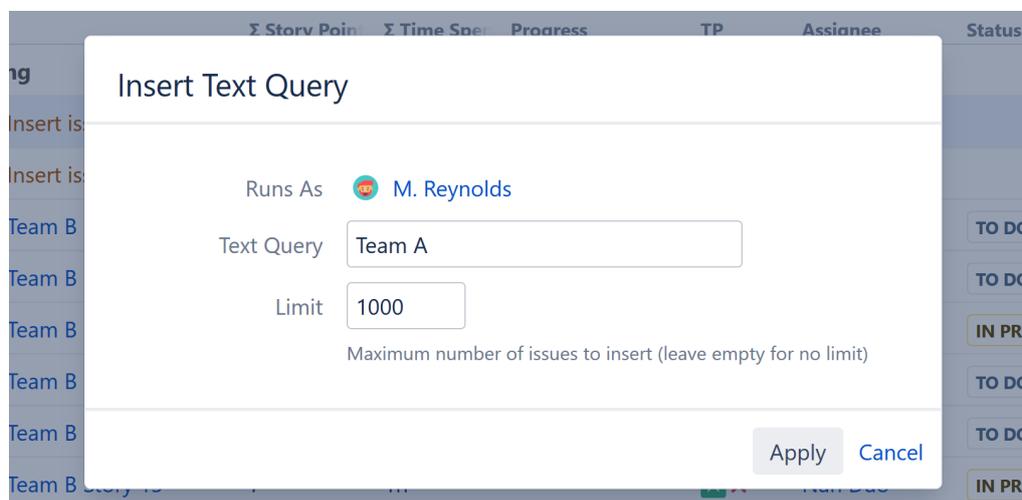
### When to Use the Structure Inserter

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.

## 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.

Key	Summary	Σ Story Point	Σ Time Spent	Progress	TP	Assignee	Status
	Text Query Inserter	207	3d 2h	<div style="width: 100%;"></div>			
	+ Insert issues: Team A						
STMA-21	Team A Story 20			<div style="width: 100%;"></div>		Unassigned	TO DO
STMA-16	Team A Story 16	15		<div style="width: 100%;"></div>		Unassigned	TO DO
STMA-15	Team A Story 15	6		<div style="width: 100%;"></div>		Unassigned	TO DO
STMA-14	Team A Story 14	13		<div style="width: 100%;"></div>		Unassigned	TO DO

## 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



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.

## When to Use the Filter Generator

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

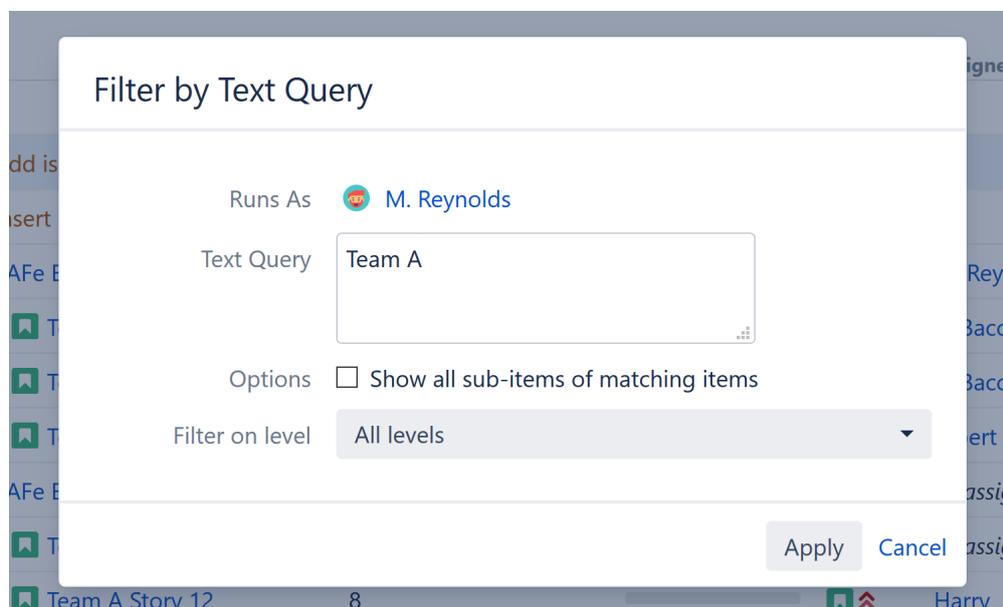
It is particularly useful when used in conjunction with other generators, in order to limit the number of issues based on a specific Jira or Structure attribute – in the example above, we've filtered out everything except unassigned issues.



Use the Filter generator when you always want issues filtered within the structure. To temporarily filter issues, use the [Filter](#) button in the toolbar or [Quick Transformations](#).

## Configuring a Filter

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



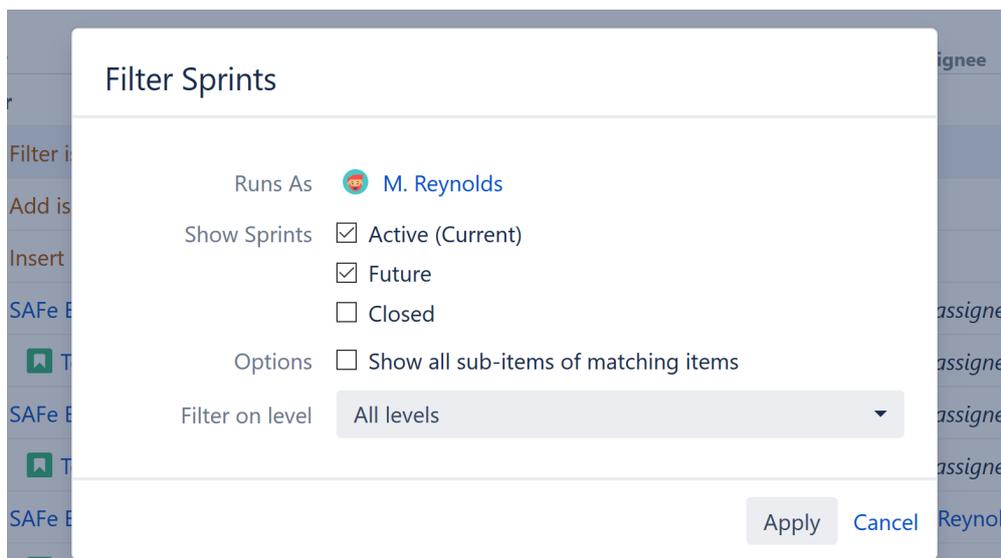
While each filter has its own options, they all include the following:

- **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](#) to learn more about customizing levels.

## 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.

- i Filtering by Sprint (and the predefined Hide Closed Sprints) is limited to inserted structure(s), 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.

## Insertor/Extender Duplicates Filter

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

Here's how this can happen:

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](#) 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:

## Project Dependencies ▾

Summary

 Project Dependencies

 Add issues linked by **Blocks**: parent is blocked by children

---

**+** Insert issues: **Project = Demo**

---

▾  Story 1

---

    ▾  Story 2

---

 Story 3

---

 Story 4

---

▾  Story 2

---

 Story 3

---

 Story 4

---

 Story 3

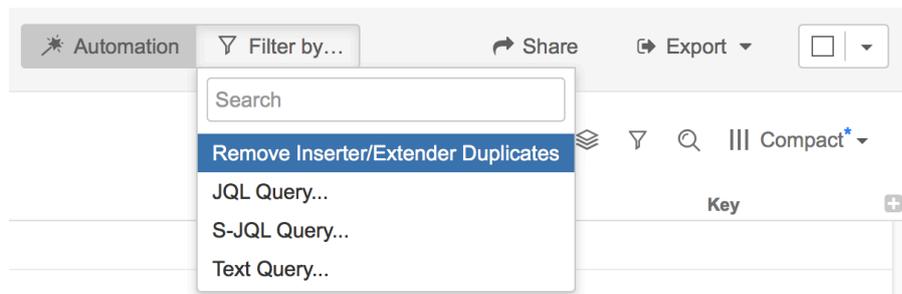
---

 Story 4

---

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:



The screenshot shows the top toolbar of the application. The 'Automation' button is on the left, followed by the 'Filter by...' dropdown menu which is open. The dropdown menu contains a search input field and several options: 'Remove Inserter/Extender Duplicates' (highlighted in blue), 'JQL Query...', 'S-JQL Query...', and 'Text Query...'. To the right of the dropdown are 'Share' and 'Export' buttons. Below the toolbar, a table header is visible with a 'Key' column and a '+' icon.

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

## ☰ Project Dependencies ▾

Summary

 **Project Dependencies**

---

⚠ Remove Inserter/Extender Duplicates

---

🔗 Add issues linked by **Blocks**: parent **is blocked by** children

---

+ Insert issues: **Project = Demo**

---

▾  Story 1

---

▾  Story 2

---

 Story 3

---

 Story 4

---

---

### 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:

☰ Project Dependencies ▾

Summary

☰ Project Dependencies

🔗 Add issues linked by **Blocks**: parent is blocked by children

+ Insert issues: **Project = Demo**

- ▾ 📄 Story 1
  - ▾ 📄 Story 2
    - 📄 Story 4
  - ▾ 📄 Story 3
    - 📄 Story 4
- ▾ 📄 Story 2
  - 📄 Story 4
- ▾ 📄 Story 3
  - 📄 Story 4
- 📄 Story 4

With the filter applied, any identical instances are removed:

☰ Project Dependencies ▾

Summary

☰ Project Dependencies

⚡ Remove Inserter/Extender Duplicates

🔗 Add issues linked by **Blocks**: parent is blocked by children

+ Insert issues: **Project = Demo**

- ▾ 📄 Story 1
  - ▾ 📄 Story 2
    - 📄 Story 4
  - ▾ 📄 Story 3
    - 📄 Story 4

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:

☰ Project Dependencies ▾

Summary

☀ ☰ Project Dependencies

🔗 Add issues linked by **Blocks**: parent is **blocked by** children

+ Insert issues: **Project = Demo**

▾ 📄 Story 1

▾ 📄 Story 2

🔄 Story 1

▾ 📄 Story 2

▾ 📄 Story 1

🔄 Story 2

📄 Story 3

📄 Story 4

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

## ☰ Project Dependencies ▾

Summary

- ☠ ☰ Project Dependencies
- ⚑ Remove Inserter/Extender Duplicates
- 🔗 Add issues linked by **Blocks**: parent is **blocked by** children
- + Insert issues: **Project = Demo**
- ▾ 📄 Story 2
  - ▾ 📄 Story 1
    - ↻ Story 2
- 📄 Story 3
- 📄 Story 4

## 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.

## Customize Your Sort

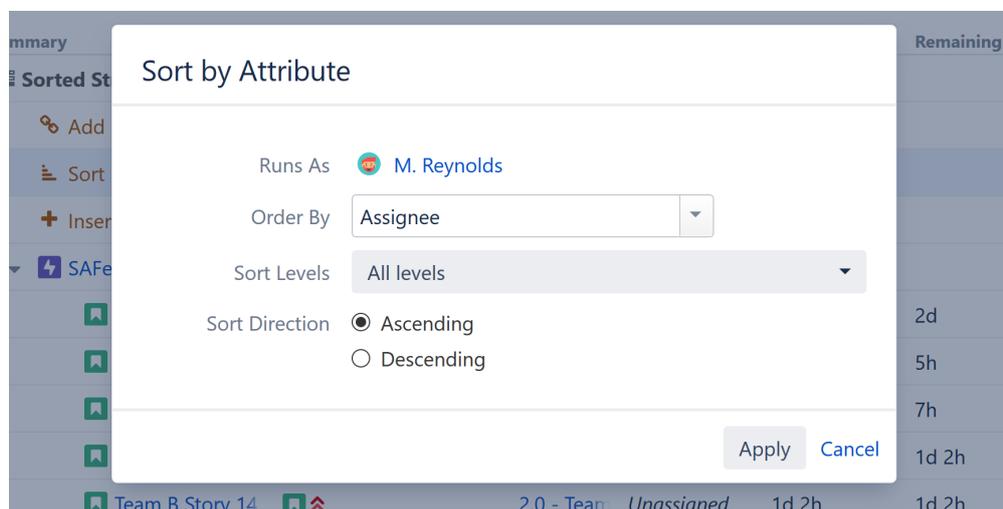
Most of the options under the "Sort By..." Automation are applied the moment they are selected, and you will not be asked to set parameters for the sort. However, you can still customize a Sort generator by locating it within your structure and double-clicking its summary.

Sorted Structure ▾

☆ 🏠 🔍 📄 Planning ▾

Key	Summary	TP	Due Date	Fix Version	Assignee	Original Estimate	Remaining Estimate	Σ Remaining Estimate
Sorted Structure								19w 1d 6h
Add issues belonging								
Sort by Assignee								
Insert issues: Project								
SPR-9	SAFE Epic 9				Bob			1w 1d
STMB-4	Team B Story 4				Albert	2d	2d	2d
STMB-9	Team B Story 9				Bob	5h	5h	5h
STMB-7	Team B Story 7				M. Reynolds	1d 7h	7h	7h
STMB-6	Team B Story 6			1.0 - Team	Nah Duo	1d 2h	1d 2h	1d 2h

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**.

## 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** issues from all active projects (and use **Extenders** 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](#) 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.

## Rank Sort

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

## Manual Reordering Generator

If your structure was built using a [JQL Query Inserter](#), 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](#).

## Group Generators

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

☰ Grouped Structure ▾ 🏠 ⚙️ 🔍 🔍 🔍 Basic view\* ▾

Key	Summary	Σ Story Point	Σ Time Spent	Progress	TP	Assignee	Status
☰ Grouped Structure		207	3d 2h	<div style="width: 50%;"></div>			
📦 Group by Assignee							
+ Insert issues: project = "SAF"							
albert	Albert	15		<div style="width: 50%;"></div>			
STMA-8	Team A Story 8	15		<div style="width: 50%;"></div>	📌 ⬆️	Albert	TO DO
anna	Anna M.	17		<div style="width: 50%;"></div>			
STMA-11	Team A Story 11	17		<div style="width: 50%;"></div>	📌 ⬆️	Anna M.	TO DO
rleitch	Bob	21		<div style="width: 50%;"></div>			
STMA-10	Team A Story 10	12		<div style="width: 50%;"></div>	📌 ⬆️	Bob	TO DO

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:



Key	Summary	Σ Story Points	Σ Time Spent
	<b>Grouped Structure</b>	207	3d 2h
	Group by Sprint		
	Group by <b>Assignee</b>		
	Insert issues: <b>project = "SAF"</b>		
	Active sprint: Team A Sprint 1 27		1d 3h

## 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.

Key	Summary	Σ Story Points	Σ Time Spent	Progress
	<b>Grouped Structure</b>	393	1w 2d 4h	
	Add issues belonging to epic			
	Group by <b>Assignee</b>			
	Insert epics from " <b>SAFe Prog</b>			
albert	Albert	82	4d	
SPR-9	SAFe Epic 9	54	1d	

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 by Issue Field**

Runs As M. Reynolds

Issue Field **Assignee** ▼

Group on level **Current level** ▼

Consider other groups  
Group items created by other groupers are normally skipped, unless this option is enabled.

Allow changes via Structure

Apply Cancel

- **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](#) 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 text attributes 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](#)).

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.

## 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
- **Text attributes:** built-in and custom text fields
- **Portfolio parent link:** as defined in 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 Desk



It is not possible to group by date or numbers.

## 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:

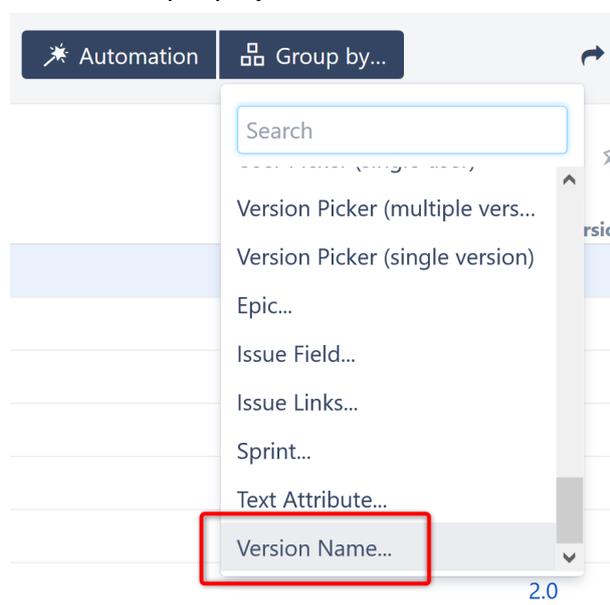
- **Moving from one group to another** - This will remove the original value and add the new value.

- **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.

## 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.

Grouped by Version Name ▾

☆ ☰ 🔍 Basic view\* ▾

Key	Summary	Fix Version/s	Progress	TP
Grouped by Version Name				
Group by Fix Version/s name				
1.0 - Public (DP), Public Release (TWP)				
✓ DP-1	▶ Story DP1	1.0, 1.5	<div style="width: 100%; height: 10px; background-color: green;"></div>	▶ =
✓ TWP-3	▶ Story TWP1	1.0	<div style="width: 100%; height: 10px; background-color: green;"></div>	▶ =
1.5				
✓ DP-1	▶ Story DP1	1.0, 1.5	<div style="width: 100%; height: 10px; background-color: green;"></div>	▶ =
DP-2	▶ Story DP2	1.5	<div style="width: 100%; height: 10px; background-color: green;"></div>	▶ =
DP-4	▶ Story DP4	1.5	<div style="width: 100%; height: 10px; background-color: gray;"></div>	▶ =
2.0 - New Layout				
✓ DP-3	▶ Story DP3	2.0	<div style="width: 100%; height: 10px; background-color: green;"></div>	▶ =
TWP-4	▶ Story TWP2	2.0	<div style="width: 100%; height: 10px; background-color: gray;"></div>	▶ =



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).

The screenshot shows a software interface with a 'Group by...' dropdown menu. The menu is open, displaying various options. The 'Fix Version/s' option is highlighted with a red box. The background shows a table with columns for 'Key', 'Summary', and 'Fix Version/s', with rows grouped by version name.

Key	Summary	Fix Version/s
Grouped by Version Name		
Group by Fix Version/s		
2.5 - Additional Features		
TWP-5	Story TWP3	
2.0 - New Layout		
TWP-4	Story TWP2	
1.0 - Public Release		
TWP-3	Story TWP1	1.0
2.5 - Additional Features		
DP-12	Story DP5	2.5
2.0		
DP-3	Story DP3	2.0

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.

## Group by Tempo Account

If you're using the [Tempo Timesheets app](#), our integration allows you to track all your KPIs per account within a structure.

The screenshot shows a software interface with a table of tasks grouped by Tempo Account. The table has columns for 'Key', 'Summary', 'Progress', and 'Σ Time Spent'. The tasks are grouped into three accounts: Account A, Account B, and ClosedAc (Closed).

Key	Summary	Progress	Σ Time Spent
Account A			
SCRUM-300	Task A		1d 6h
SCRUM-298	Sub-task 1		1d 2h
			4h
Account B			
SCRUM-140	Task K		3d 3h
SCRUM-10	Story 2		7h
			2d 4h
ClosedAc (Closed)			
SCRUM-295	Sub-task 2		4d 2h
SCRUM-291	Story X		1d 6h
			2d 4h

## Grouping by Tempo Accounts

1. Add tasks, either manually or using [Structure Automation](#).
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.

#### 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.

## Extend Generators

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

☰ Top-Down Automation ▾ ☆ ⚙️ 🔍 ||| Basic view ▾

Key	Summary	Progress	Status	Assignee	Icons
	 <b>Top-Down Automation</b>	<div style="width: 50%;"><div style="background-color: #4CAF50; height: 10px;"></div></div>			
	 Add issues linked by <b>Blocks</b> : parent blocks c				
	 Add issues belonging to epics				
	 Insert epics from "SAFe Team A Scrum"				
SPR-2	▾  SAFe Epic 2	<div style="width: 10%;"><div style="background-color: #4CAF50; height: 10px;"></div></div>	BACKLOG	M. Reynolds	 
STMA-10	▾  Team A Story 10	<div style="width: 10%;"><div style="background-color: #4CAF50; height: 10px;"></div></div>	TO DO	C. Bacca	 
STMA-13	 Team A Story 13	<div style="width: 10%;"><div style="background-color: #4CAF50; height: 10px;"></div></div>	TO DO	Unassigned	 
STMA-9	▾  Team A Story 9	<div style="width: 10%;"><div style="background-color: #4CAF50; height: 10px;"></div></div>	TO DO	C. Bacca	 
STMA-3	 Team A Story 3	<div style="width: 10%;"><div style="background-color: #4CAF50; height: 10px;"></div></div>	IN PROGRESS	Jack Brown	 
STMA-8	 Team A Story 8	<div style="width: 10%;"><div style="background-color: #4CAF50; height: 10px;"></div></div>	TO DO	Albert	 
SPR-1	▾  SAFe Epic 1	<div style="width: 10%;"><div style="background-color: #4CAF50; height: 10px;"></div></div>	SELECTED FOR DEVELOPMENT	M. Reynolds	 

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

## Types of Extenders

The following Insert generators are available:

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

## 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.

## 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.

☰ Top-Down Automation ▾ 🔍 🏠 🔍 📄 Basic view ▾

Key	Summary	Progress	Status	Assignee	Icons
☰	☰ Top-Down Automation	<div style="width: 50%; background-color: #4CAF50;"></div>			
	🔗 Add issues linked by <b>Blocks</b> : parent blocks				
	➕ Insert issues from "SAFe Team A Scrum"				
STMA-1	📄 Team A Story 1	<div style="width: 50%; background-color: #4CAF50;"></div>	IN PROGRESS	C. Bacca	📄 ⬆️
STMA-15	📄 Team A Story 15	<div style="width: 0%; background-color: #4CAF50;"></div>	TO DO	Unassigned	📄 ⬆️
STMA-2	📄 Team A Story 2	<div style="width: 50%; background-color: #4CAF50;"></div>	IN PROGRESS	Jack Brown	📄 ⬆️
STMA-14	📄 Team A Story 14	<div style="width: 0%; background-color: #4CAF50;"></div>	TO DO	Unassigned	📄 ⬆️
STMA-8	📄 Team A Story 8	<div style="width: 0%; background-color: #4CAF50;"></div>	TO DO	Albert	📄 ⬆️
STMA-15	📄 Team A Story 15	<div style="width: 0%; background-color: #4CAF50;"></div>	TO DO	Unassigned	📄 ⬆️

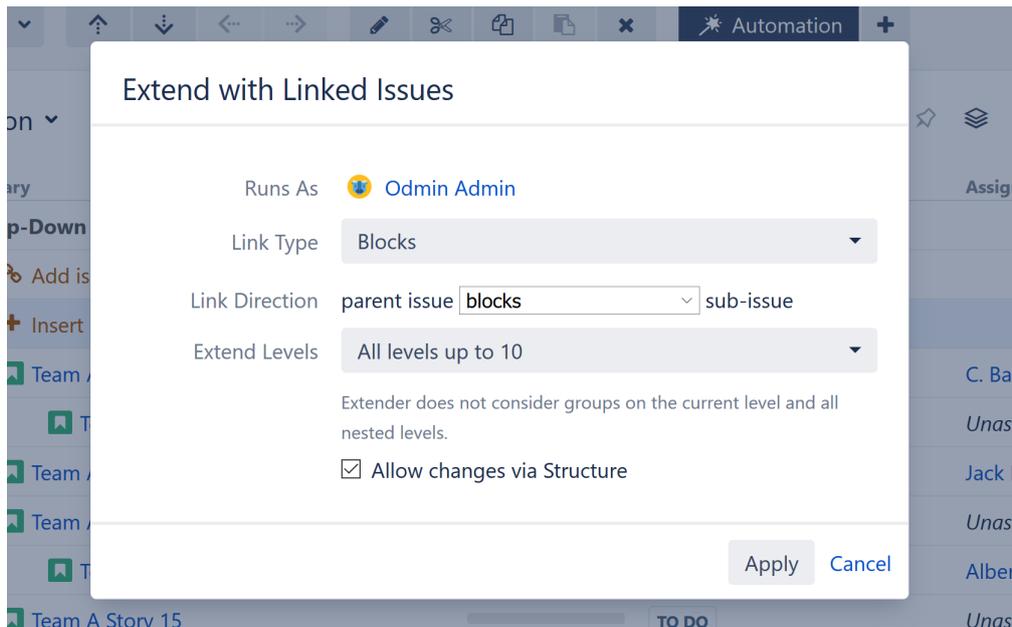


When using the Linked Issues Extender, some issues may appear in your structure more than once. In the example above, "Team A Story 15" appears twice, because it met the criteria for the original Inserter AND it was linked to "Team A Story 1."

You can use the [Inserter/Extender Duplicates Filter](#) 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](#) 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.

## 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.

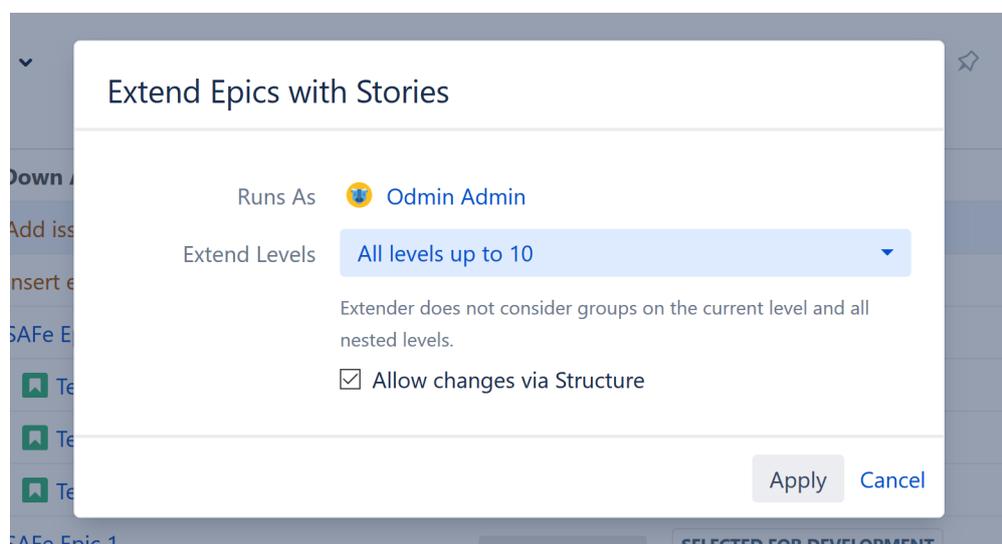
☰ Top-Down Automation ▾ ☆ ⌵ 🔍 ||| Basic view ▾

Key	Summary	Progress	Status	Assignee	Icons
☰ Top-Down Automation					
🔗 Add issues belonging to epics					
+ Insert epics from "SAFe Team A Scrum"					
SPR-2 ▾	SAFe Epic 2	<div style="width: 100%;"></div>	BACKLOG	M. Reynolds	🔗 ⬆️
STMA-10	Team A Story 10	<div style="width: 100%;"></div>	TO DO	C. Bacca	🔗 ⬆️
STMA-9	Team A Story 9	<div style="width: 100%;"></div>	TO DO	C. Bacca	🔗 ⬆️
STMA-8	Team A Story 8	<div style="width: 100%;"></div>	TO DO	Albert	🔗 ⬆️
SPR-1 ▾	SAFe Epic 1	<div style="width: 100%;"></div>	SELECTED FOR DEVELOPMENT	M. Reynolds	🔗 ⬆️
STMA-13	Team A Story 13	<div style="width: 100%;"></div>	TO DO	Unassigned	🔗 ⬆️
STMA-12	Team A Story 12	<div style="width: 100%;"></div>	TO DO	Harry	🔗 ⬆️

In the example above, we built our hierarchy by adding all our epics with an [Agile Board Inserter](#), 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](#) 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.

## 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.

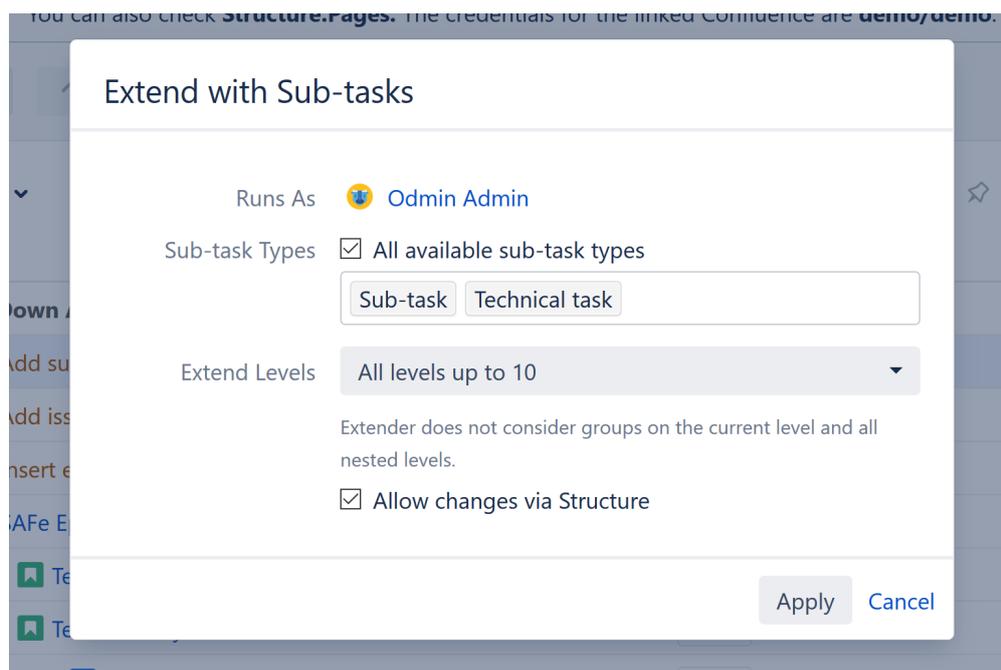
Key	Summary	Progress	Status	Assignee	Icons
	☰ Top-Down Automation	<div style="width: 50%;"></div>			
	🔗 Add sub-tasks				
	🔗 Add issues belonging to epics				
	➕ Insert epics from "SAFe Team A Scrum"				
SPR-2	☑ SAFe Epic 2	<div style="width: 100%;"></div>	BACKLOG	M. Reynolds	🔗 ⬆
STMA-10	👤 Team A Story 10	<div style="width: 100%;"></div>	TO DO	C. Bacca	👤 ⬆
STMA-9	☑ Team A Story 9	<div style="width: 100%;"></div>	TO DO	C. Bacca	👤 ⬆
STMA-20	👤 Sub-task 4	<div style="width: 100%;"></div>	TO DO	C. Bacca	👤 ⬆
STMA-8	☑ Team A Story 8	<div style="width: 100%;"></div>	TO DO	Albert	👤 ⬆
STMA-17	👤 Sub-task 1	<div style="width: 100%;"></div>	IN PROGRESS	H. Solo	👤 ⬆
STMA-18	👤 Sub-task 2	<div style="width: 100%;"></div>	IN PROGRESS	Albert	👤 ⬆

In the example above, we built our hierarchy by:

1. Adding all our epics with [Agile Board Inserter](#)
2. Placing our issues beneath them, using the [Stories Under Epics Extender](#)
3. Placing sub-tasks beneath issues, using the Sub-tasks Extender

## 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](#) 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.

## Child Issues for Portfolio Extender

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

## Customize Your Extender

Each Child Issues (Portfolio) 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:

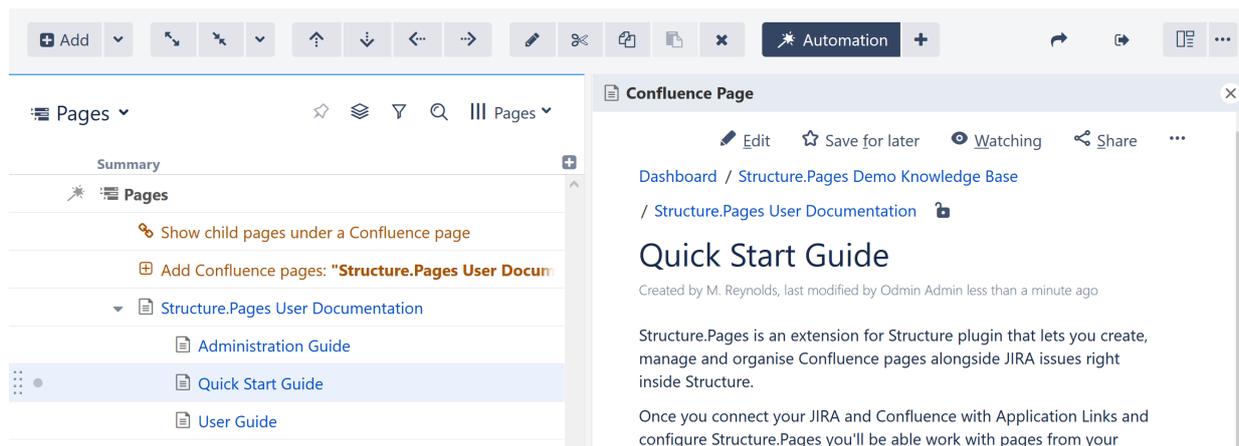
- *All levels up to 10* (default) - The extender will be applied to the first 10 levels of the hierarchy, starting from the level where the extender itself is located.
- *All levels* - The extender will be applied to the current level and all its descendants.
- *Current level only* - The extender will 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](#) 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.

## Pages Extenders

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



## Child Pages Extender

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

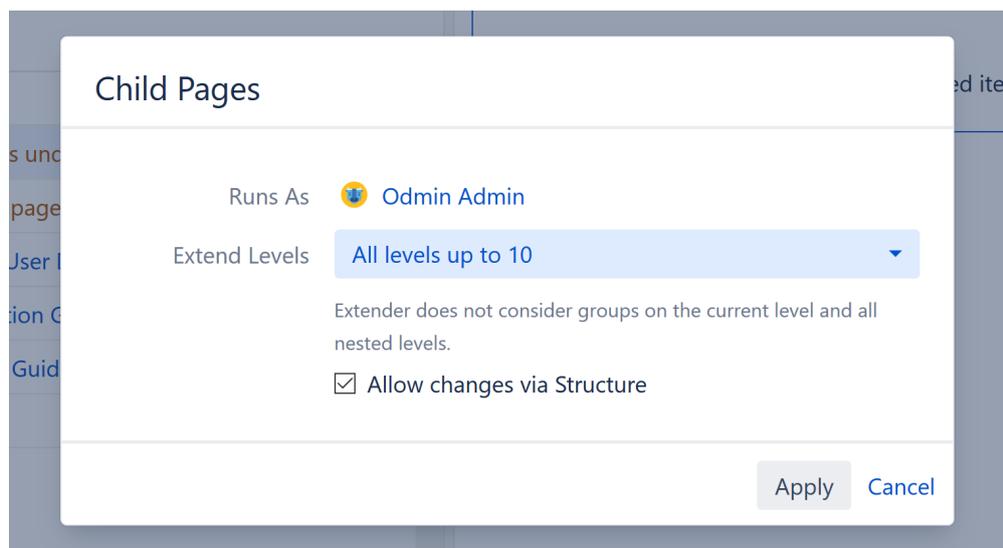
✔ You can add pages manually, or using the [Insert](#) generator or Linked Pages Extender.

## 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](#) 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.

## 2.3.6 Effectors

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

### 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.

Key	Summary	Status	Progress	Story Points	Σ Story Points
WNP-9	WNP-9				51
	⚡ Epic Story Points Effector				
WNP-9	Epic 1	IN PROGRESS			23
WNP-1	Story 1	IN PROGRESS		5	5
WNP-3	Story 3	IN PROGRESS		10	10
WNP-2	Story 2	DONE		8	8

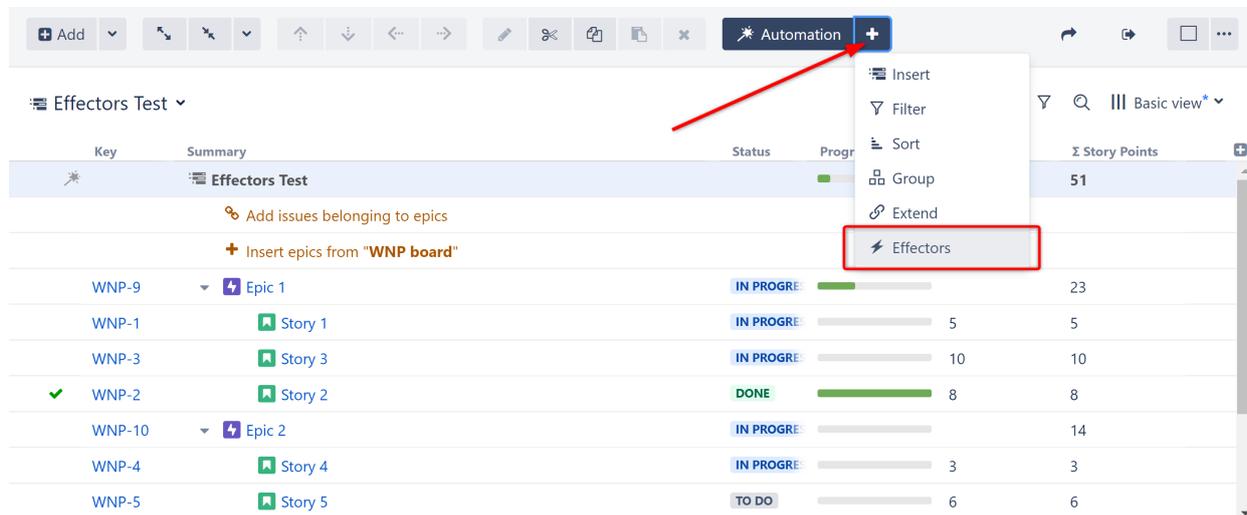
Structure with Effector



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 Story Points for Epics and Write the Values to Jira](#) article.

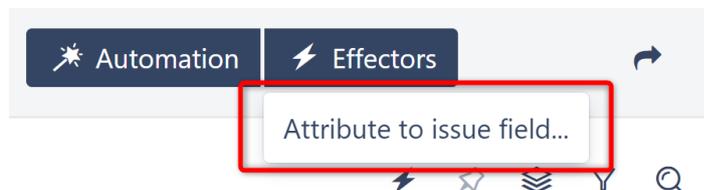
## Adding Effectors to a Structure

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



Selecting Effectors in the Automation Menu

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 source attribute (with the values you want to write to Jira)
- Set Value To: 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

Save Attribute Value to Issue Field

Name **Attribute to issue field Assignee**

Get Value From\* Select attribute...

Set Value To\* Assignee

Limit To Enter JQL to apply the effector only to matching issues in the structure

Send email notifications

Save and Run Save Cancel

Attribute to Issue Field Effector Settings

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.

## Running an Effector

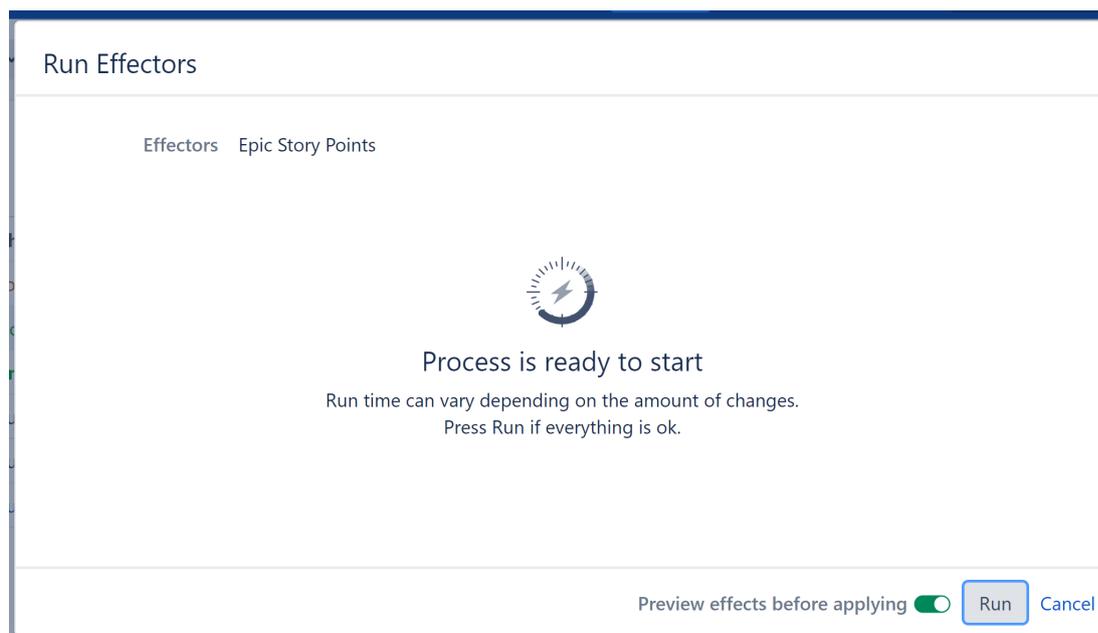
Now that you've created an Effector, you need to run it for any changes to take effect. To do that, locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.

Key	Summary	Status	Progress	Story Points	Σ Story Points
	Effectors Test				51
	Add issues belonging to epics				
	Epic Story Points				
	Insert epics from "WNP board"				
WNP-9	Epic 1	IN PROGRE			23
WNP-1	Story 1	IN PROGRE		5	5
WNP-3	Story 3	IN PROGRE		10	10

Run an Effector from the Structure Board

## 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.



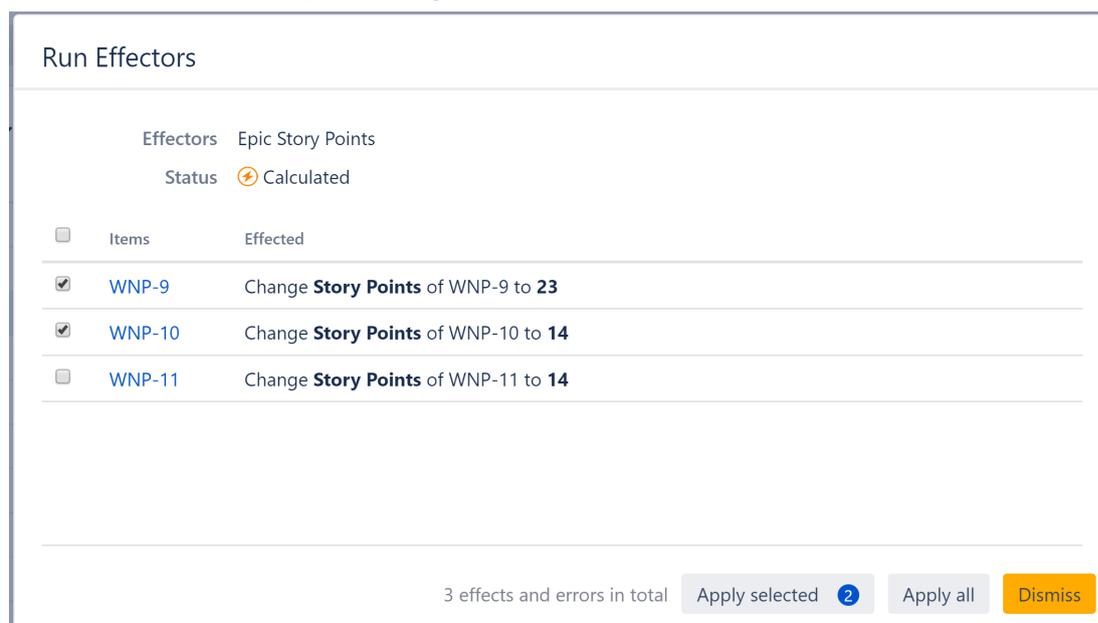
Run Effector Dialogue



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.

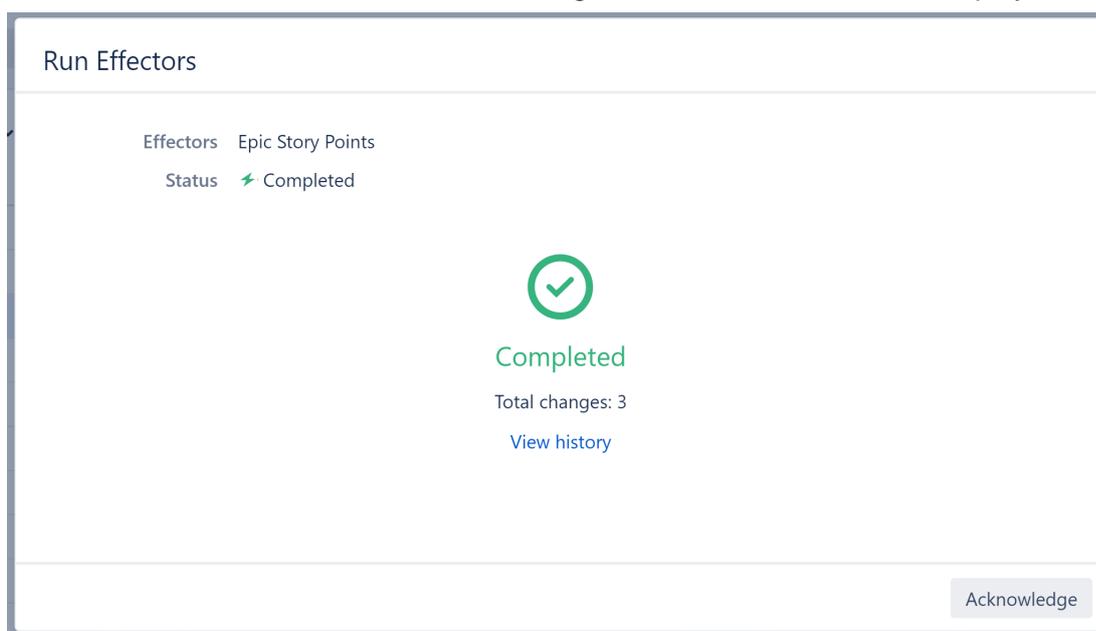


Preview changes that will be made by the Effector

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

**i** 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.



Effectors Completed Confirmation

## Review Changes Made by an Effector

To review the changes made by an Effector, click **View history** on the Completed dialogue screen.

## Effector process audit

Effectors Epic Story Points  
 Started 27/Jan/20 6:06 PM  
 Finished 27/Jan/20 6:06 PM  
 Status ✔ Completed  
 Structure [Effectors Test](#)



<input type="checkbox"/> Items	Reverse action	Effect
<input type="checkbox"/> <a href="#">WNP-11</a>	Clear <b>Story Points</b> value of WNP-11	Changed <b>Story Points</b> of WNP-11 to <b>14</b> <a href="#">Revert</a>
<input type="checkbox"/> <a href="#">WNP-10</a>	Clear <b>Story Points</b> value of WNP-10	Changed <b>Story Points</b> of WNP-10 to <b>14</b> <a href="#">Revert</a>
<input type="checkbox"/> <a href="#">WNP-9</a>	Clear <b>Story Points</b> value of WNP-9	Changed <b>Story Points</b> of WNP-9 to <b>23</b> <a href="#">Revert</a>

3 records in total

 0


Review Effector history

 If you've already closed the Completed dialogue, you can also reach this information from the [Manage Effectors](#) screen.

## Revert Changes

If you need to revert any changes made by the Effector, click the **Revert** link next to the change. To revert multiple changes, select each and click **Revert Selected**, or click **Revert All**.

## Example Use Cases for the Attribute to Issue Field Effector

Some popular uses for this Effector include:

- [Add story points for epics](#)
- [Update issue assignees based on parent issues](#)
- [Calculate the cost to complete issues and projects, and write those cost to a Jira field](#)
- 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).

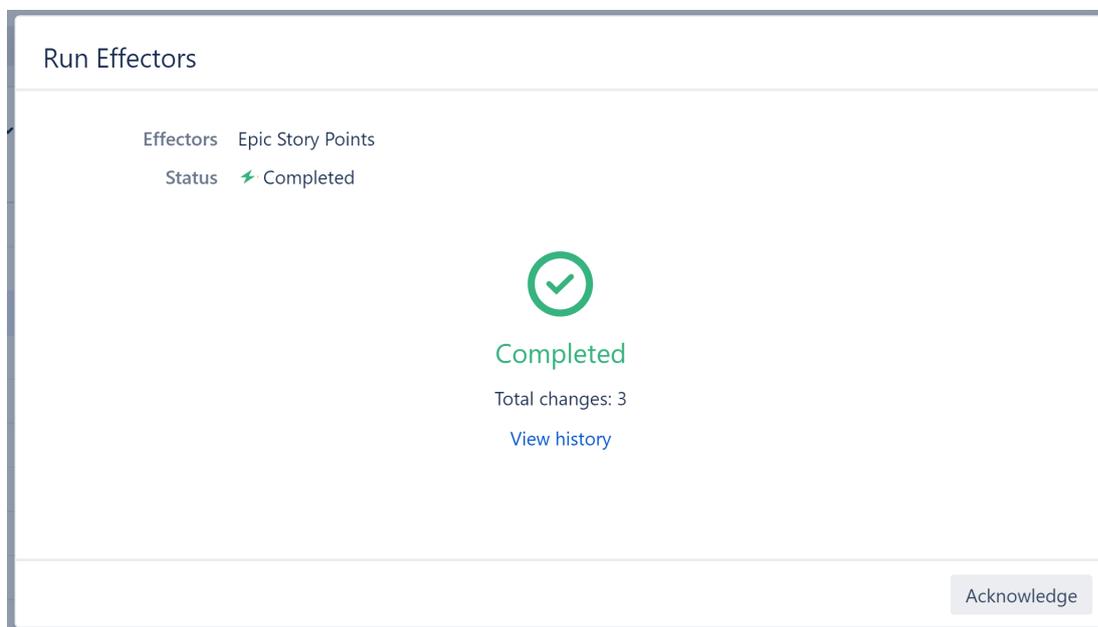
## Revert Effector Changes

You can revert changes made by an Effector:

- [Immediately after running the Effector](#)
- [At any time, using the Manage Effectors page](#)
- You can even [revert changes made during a revert process](#)

## 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.



Effectors Completed Confirmation

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**.

## Effector process audit

Effectors Epic Story Points  
 Started 27/Jan/20 6:06 PM  
 Finished 27/Jan/20 6:06 PM  
 Status ⚡ Completed  
 Structure Effectors Test



<input type="checkbox"/>	Items	Reverse action	Effect
<input type="checkbox"/>	WNP-11	Clear <b>Story Points</b> value of WNP-11	Changed <b>Story Points</b> of WNP-11 to <b>14</b> <a href="#">Revert</a>
<input type="checkbox"/>	WNP-10	Clear <b>Story Points</b> value of WNP-10	Changed <b>Story Points</b> of WNP-10 to <b>14</b> <a href="#">Revert</a>
<input type="checkbox"/>	WNP-9	Clear <b>Story Points</b> value of WNP-9	Changed <b>Story Points</b> of WNP-9 to <b>23</b> <a href="#">Revert</a>

3 records in total

 0

Review Effector history

## 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 screenshot shows the Jira Software interface. The 'Structure' menu is open, and 'Manage Effectors' is highlighted with a red box. A red arrow points from the 'Manage Effectors' option to the 'Structure' menu. The background shows a table with columns for Assignee, Remaining Estimate, Remaining Cost, and Finish Cost.

Manage Effectors in the Structure Menu

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

**Manage Effectors**

Manage Effectors ?

Effector name  Structure ▼ Status ▼ Run by ▼ Clear

ID	Effectors	Structure	Run By	Started	Finished	Status	Actions
66	<b>Due Date</b> to issue field <b>Finish Date</b>	Gantt Chart Demo	M. Reynolds	24/Mar/20 3:09 ...	24/Mar/20 3:09 ...	✔ Completed	Revert
65	<b>Attribute</b> to issue field <b>Start Date</b>	Gantt Chart Demo	M. Reynolds	24/Mar/20 3:08 ...	24/Mar/20 3:08 ...	✔ Completed	Revert
33	<b>Σ Formula</b> to issue field <b>FinishCost</b>	Cost Analysis	M. Reynolds	13/Mar/20 11:25...	13/Mar/20 11:25...	✔ Completed	Revert

Manage Effectors screen

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.

Manage Effectors

Effector name  Structure ▼ Status ▼

ID	Effectors	Structure	Run By	Started
72	Revert of 71	Gantt Chart Demo	M. Reynolds	24/Mar/20 4:42 ...
71	<b>Start date</b> to issue field <b>Start Date</b>	Gantt Chart Demo	M. Reynolds	24/Mar/20 4:29 ...
70	Same Assignee as Parent	Gantt Chart Demo	M. Reynolds	24/Mar/20 4:29 ...

Revert process listed on the Effector Processes screen

## 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**.

The screenshot shows the Jira Software interface. The top navigation bar includes 'Dashboards', 'Projects', 'Issues', 'Boards', 'Structure', and '+ Create'. A red arrow points to the 'Structure' dropdown menu, which is open and shows several options. The 'Manage Effectors' option is highlighted with a red box. Below the menu, a table displays structure data with columns for Assignee, Remaining Estimate, Remaining Cost, and Finish Cost.

Assignee	Remaining Estimate	Remaining Cost	Finish Cost
55	2d	880	660
		5270	5,270
60	2d 4h	1410	1200, 1,410
70	3h	210	210
65	1d 2h	3050	650, 3,050
75	4d	2400	2,400
55	1d 4h	810	660, 810
50	3h	150	150

Manage Effectors in the Structure menu

**i** 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.

The screenshot shows the footer of the Structure Board. It includes a 'TO DO' section with a progress bar. Below this, there is a link for 'Effectors' with a green progress bar indicating 100% completion. A red arrow points to the 'Effectors' link.

**✓** 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** 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.

- Click **Revert** 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.

Manage Effectors							
Manage Effectors <span style="float: right;">?</span>							
Effector name <input type="text"/>		Structure <span>▼</span>	Status <span>▼</span>	Run by <span>▼</span>	Clear		
ID	Effectors	Structure	Run By	Started	Finished	Status	Actions
66	<b>Due Date</b> to issue field <b>Finish Date</b>	Gantt Chart Demo	M. Reynolds	24/Mar/20 3:09 ...	24/Mar/20 3:09 ...	⚡ Completed	Revert
65	<b>Attribute</b> to issue field <b>Start Date</b>	Gantt Chart Demo	M. Reynolds	24/Mar/20 3:08 ...	24/Mar/20 3:08 ...	⚡ Completed	Revert
33	<b>Σ Formula</b> to issue field <b>FinishCost</b>	Cost Analysis	M. Reynolds	13/Mar/20 11:25...	13/Mar/20 11:25...	⚡ Completed	Revert

Manage Effectors screen

## 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

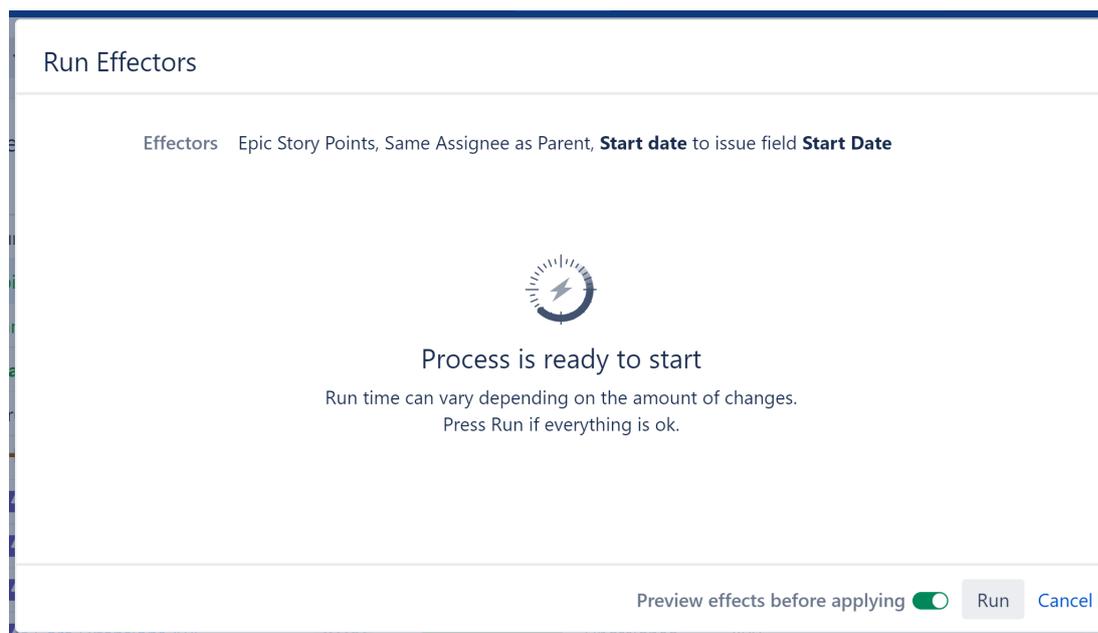
## Running All Effectors in a Structure

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:

Structure with Multiple Effectors <span>▼</span>									
Key	Summary	Status	Progress	Assignee	WSJF (Basic)	Work Logged - M.	Σ Story Poin	Parent Assignee	
Structure with Multiple Effectors									
	⚡ Epic Story Points								...
	⚡ Same Assignee as Parent								
	⚡ Start date to issue field Start Date								
Structure									
	+ Insert issues: project = "Struct								
STR-4	⚡ Non-Jira Notes Field / Column	TO DO		Unassigned	176				
✓ STR-3	⚡ Formulas	DONE		Unassigned					

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.



Run dialogue for multiple effectors

Once the changes have been applied, you can review the operation on the [Manage Effectors](#) screen and, if necessary, [revert any changes](#).

Manage Effectors ?

ID	Effectors	Structure	Run By	Started	Finished	Status	Actions
74	Epic Story Points, Same Assignee as Parent, <b>Start date</b> to issue...	Structure with Mu...	M. Reynol...	25/Mar/20 3:12 ...	25/Mar/20 3:12 ...	✔ Completed	Revert
72	Revert of 71	Gantt Chart Demo	M. Reynol...	24/Mar/20 4:42 ...	24/Mar/20 4:42 ...	✔ Completed	Revert
71	<b>Start date</b> to issue field <b>Start Date</b>	Gantt Chart Demo	M. Reynol...	24/Mar/20 4:29 ...	24/Mar/20 4:29 ...	✔ Completed	Revert

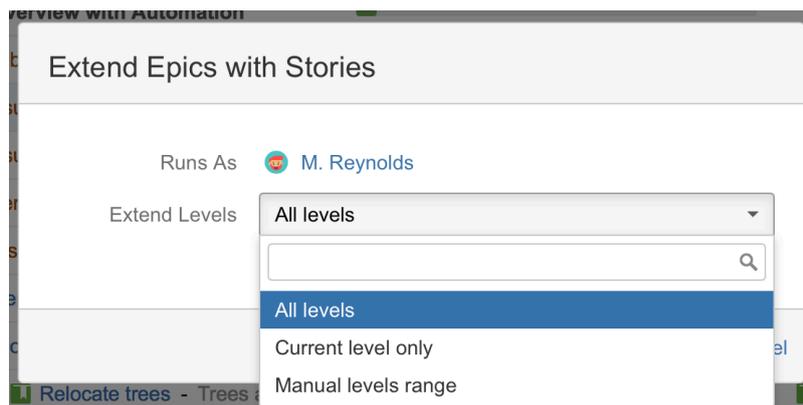
Manage Effectors - multiple effectors run at once

### 2.3.7 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.

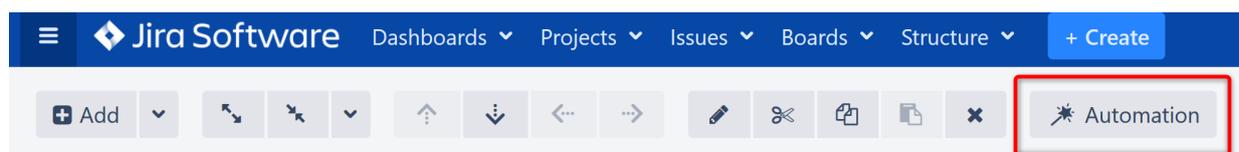
In this case, you will have three generators added under the root of the structure with the following **Levels** settings:

1. [Linked Issues Extender](#) working only on the top level - **Current level only**
2. [Stories Under Epics Extender](#) working on the second level - **Manual levels range**: from 2 to 2
3. [Sub-tasks Extender](#) 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.

### 2.3.8 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

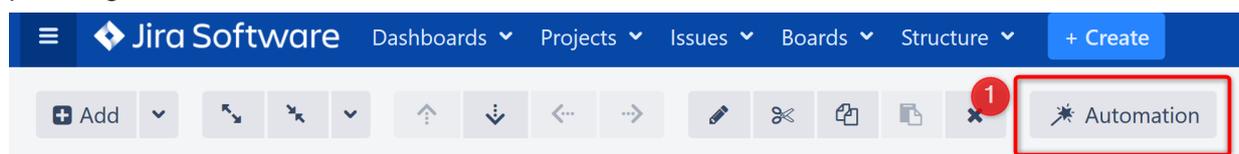
To change a generator's settings:

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).

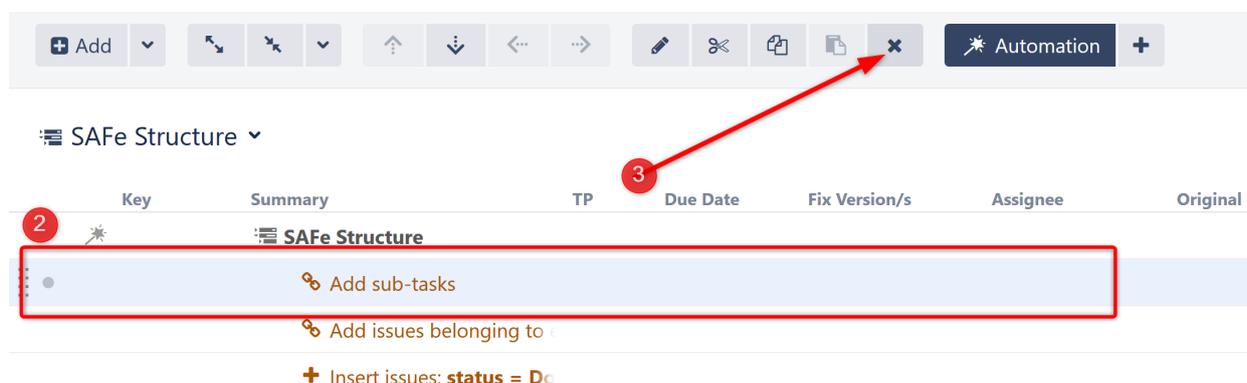
**i** To learn more about the options available for each type of generator, see the appropriate article in [Types of Generators](#).

### 2.3.9 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.



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).

### 2.3.10 Paused Generation

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.

Portfolio Overview ▾

Summary Key Time in Status Σ Time in Sta +

We couldn't load the structure. [More](#) Resume

Unresolved Assigned to me By Assignee **By Status** By Version Sub-tasks

**i** 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.

Portfolio Overview ▾

Summary Key Time in Status Σ Time in Sta +

We couldn't load the structure. [Less](#) Resume

Either the Automation or one of the applied transformations took too long to complete, so the generation has been paused. Please check the details below, review your Automation settings or enabled transformations, and try again.

You can also adjust the time limit in structure settings.

Add sub-tasks	98.46%
Insert structure <b>Portfolio Overview</b>	1.54%

Unresolved Assigned to me By Assignee **By Status** By Version Sub-tasks

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.

 Automation work was successfully resumed.

Close

## 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.

### Edit Structure

**Edit Structure** ?

Name\*

Description

Owner  

Start typing to get a list of possible matches.

Permissions User permission level is calculated by applying rules from this list, from top to bottom. **The last matching rule takes precedence.** Structure owner and Jira administrators always have **Control** permissions.

1. By default, permission level is **None**

Add Rule  to  for  [+ Add](#)

Options  Require **Edit Issue** permission on parent issue to rearrange sub-issues  
 Allow manual adjustments of generated content

Time limit  seconds (for automation)

Favorite

[Update](#) [Update and Open](#) [Back](#)

## 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.

Current	Current Structure <span>?</span>					
<a href="#">Favorite</a>	This page lets you manage your current structure.					
<a href="#">My</a>	<b>Name</b>	<b>Owner</b>	<b>Access</b>	<b>Popularity</b>	<b>Sync With</b>	<b>Operations</b>
<a href="#">Popular</a>	★ <b>Main structure</b> <b>AUTOMATION PAUSED</b>	admin	Control	1	Not synchronized <a href="#">Settings</a>	<a href="#">Configure</a> <a href="#">Views</a> <a href="#">Delete</a> <a href="#">Archive</a> <a href="#">Import</a> <a href="#">Export</a> <a href="#">Copy</a>
<a href="#">Search</a>						
<a href="#">All</a>						
<a href="#">Archived</a>						
<a href="#">Paused</a>						

 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**.

### Structure Automation Defaults

These are the default settings and thresholds for generated structures.  
Every structure has automation-related settings that override the defaults.

Default generation time limit (in seconds): 30

 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

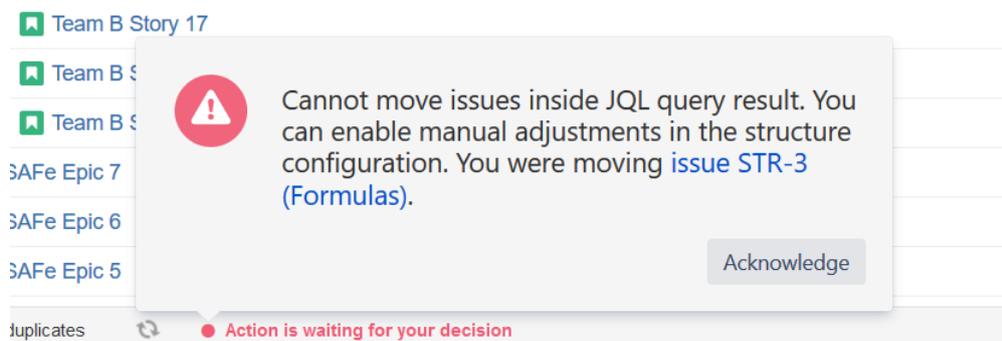
The system-wide generation time limit is initially set for 10 minutes. It can be adjusted by using Script Runner or other similar tools, or by changing the `structure.gfs.generationTimeHardLimit` property, with the [Structure Dark Feature and Fine Tuning Interface](#). The Generation Time Hard Limit 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.

### 2.3.11 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 [Automation](#) 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.



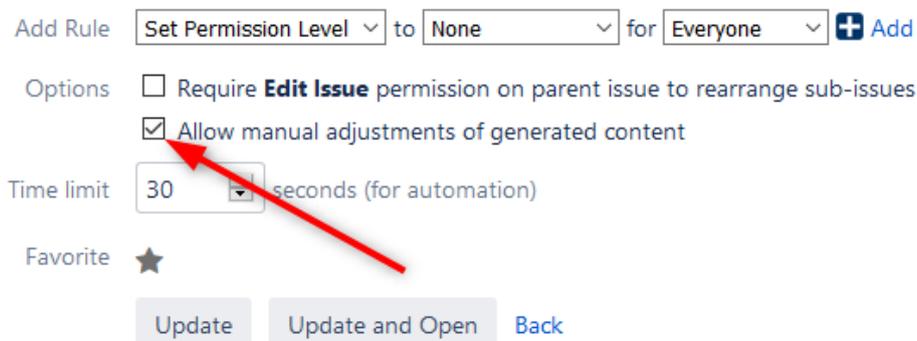
There may be times, however, when you need to move those dynamic items around freely, regardless of your generator rules. To do so, you need to enable Manual Adjustments.

## 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.

SAFe Structure

Key	Summary	Σ Story Points	Σ Time Spent	Progress	TP	Assignee
MKT-3	30-minute TV advertisement for prime-time broadcast/s	12	6h	<div style="width: 100%;"></div>		Man in Black
STMA-2	Team A Story 2	12	6h	<div style="width: 100%;"></div>		Jack Brown
MKT-2	Celebrity endorsements	5	1d 2h	<div style="width: 100%;"></div>		Bob
MKT-1	Anti-PR campaign to discredit safety of competing them	15w		<div style="width: 100%;"></div>		Man in Black
SPR-12	SAFe Epic 12	11	1d	<div style="width: 100%;"></div>		Unassigned
STMB-11	Team B Story 11	11	1d	<div style="width: 100%;"></div>		Nah Duo
SPR-11	SAFe Epic 11	15	1d	<div style="width: 100%;"></div>		M. Reynolds
STMA-1	Team A Story 1	15	1d	<div style="width: 100%;"></div>		C. Bacca

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

Automation

SAFe Structure

Key	Summary	Σ Story Points	Σ Time Spent	Progress	TP	Assignee	Status
SPR-12	SAFe Epic 12	28	3d	<div style="width: 100%;"></div>		Unassigned	BACKLOG
STMB-5	Team B Story 5	5	1d 2h	<div style="width: 100%;"></div>		Unassigned	IN PROGRESS
STMB-11	Team B Story 11	11	1d	<div style="width: 100%;"></div>		Nah Duo	IN PROGRESS
STMA-2	Team A Story 2	12	6h	<div style="width: 100%;"></div>		Jack Brown	IN PROGRESS
SPR-11	SAFe Epic 11			<div style="width: 100%;"></div>		M. Reynolds	IN PROGRESS
SPR-10	SAFe Epic 10	25	3h	<div style="width: 100%;"></div>		Unassigned	BACKLOG
STMA-6	Team A Story 6	25	3h	<div style="width: 100%;"></div>		Unassigned	IN PROGRESS

Toggle Panels menu:

- LAYOUT
  - Single Grid
  - Double Grid
  - Grid + Details
  - Grid + History
- LAYOUT OPTIONS
  - Full Screen
  - Mark Manual Adjustments
- ITEM LINK ACTION
  - Open Details
  - Navigate to Item
  - Do Nothing

## 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.

## 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](#)

## Why is Manual Adjustment Necessary?

When you use [Automation](#) 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\(s\)](#) 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](#).*



Manual Adjustment works with structures built using Automation. It does not affect structures built manually or using Synchronizers.

## 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 [Automation](#) 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.

## 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.

## 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**.

Name	Owner	Access	Popularity	Sync With	Operations
★ fgnfg	admin	Control	1	Not synchronized Settings	<a href="#">Configure</a> <a href="#">Views</a> <a href="#">Delete</a> <a href="#">Archive</a> <a href="#">Import</a> <a href="#">Export</a> <a href="#">Copy</a> <a href="#">Remove Adjustments</a>



Removing all adjustments cannot be undone.

## 2.3.12 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, they are run in the order they are listed, from top to bottom.

- This does not affect the results of Insert, Extend or Filter generators.
- This does affect the results of Group and Sort generators. With multiple Group generators, items will be grouped first by the top-most Group generator, then by the second highest, etc. The same is true with Sort - the order of the Sort generators defines which field the structure is sorted by first, second, and so on.

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.

## 2.4 Search, Filter and Transformation

The Search, Filter and Transformation features allow you to adjust a structure in order to better visualize the information you need or focus in on specific data.

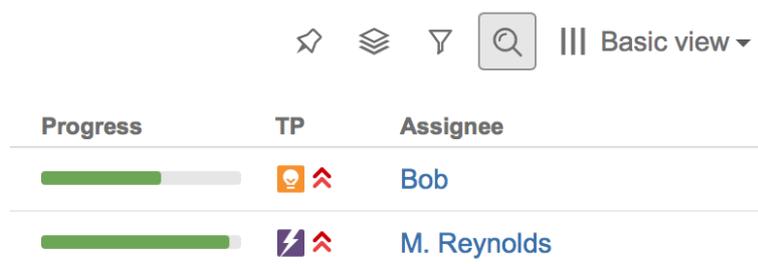
The following articles provide an overview of each feature and how to use them in your structure.

### 2.4.1 Search

The Search feature allows you to:

- Find and highlight issues in your structure
- [Filter](#) 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.

Portfolio Overview with Automation

Basic view

Key	Summary	Progress	TP	Assignee
TP-124	Site preparations	<div style="width: 100%;"></div>		Bob
SP-10	Move stuff to another place	<div style="width: 100%;"></div>		M. Reynolds
SP-5	Relocate trees - Trees are totally getting in the	<div style="width: 100%;"></div>		Unassigned
SP-13	Find the transport company	<div style="width: 100%;"></div>		M. Reynolds
SP-12	Find new location	<div style="width: 100%;"></div>		M. Reynolds
SP-16	Rent the excavation equipment	<div style="width: 100%;"></div>		M. Reynolds
SP-1	Relocate Elves - The Mountain Elves that dw	<div style="width: 100%;"></div>		Albert

Showing 55 items

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**

## Search Modes

You can search for issues within the current structure using a [Text](#), [JQL](#), or [S-JQL](#) query. To switch between these modes, click the name of the currently-selected mode and select a new mode from the menu.

Key Summary

Text status

Text

JQL

S-JQL

TTY-1

Albert

Custom Statuses

## 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	<i>structural hierarchy</i>	Looks for items that have <b>all</b> mentioned words in the <b>Summary</b> field. Each word must be present in the summary or name, or the summary must have a word that <b>begins with</b> the specified word. The words may appear in any order.
Quoted excerpt	<i>"the quick brown fox"</i>	When quotes are used, the search looks for the entire phrase in the summary or name fields.
Issue keys	<i>MARS-1, MARS-331</i>	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.

## 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](#).

## S-JQL Search

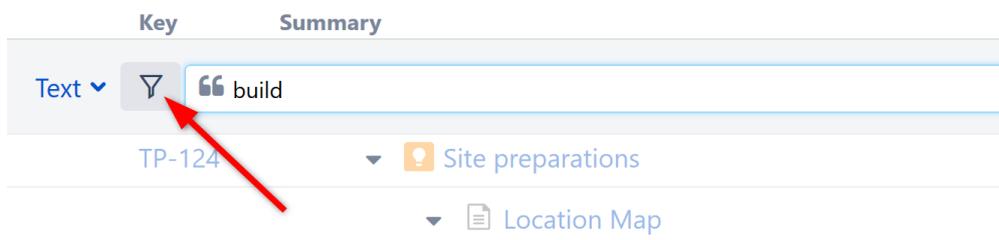
S-JQL ([Structured JQL](#)) is a special language that allows you to search for issues by their relationships in the current structure. For examples, `root` matches all top-level issues, `root` or `child of root` matches the first two levels, and `child of [priority = Critical]` matches all children of critical issues. To learn more about S-JQL conventions, see [Structured JQL](#).



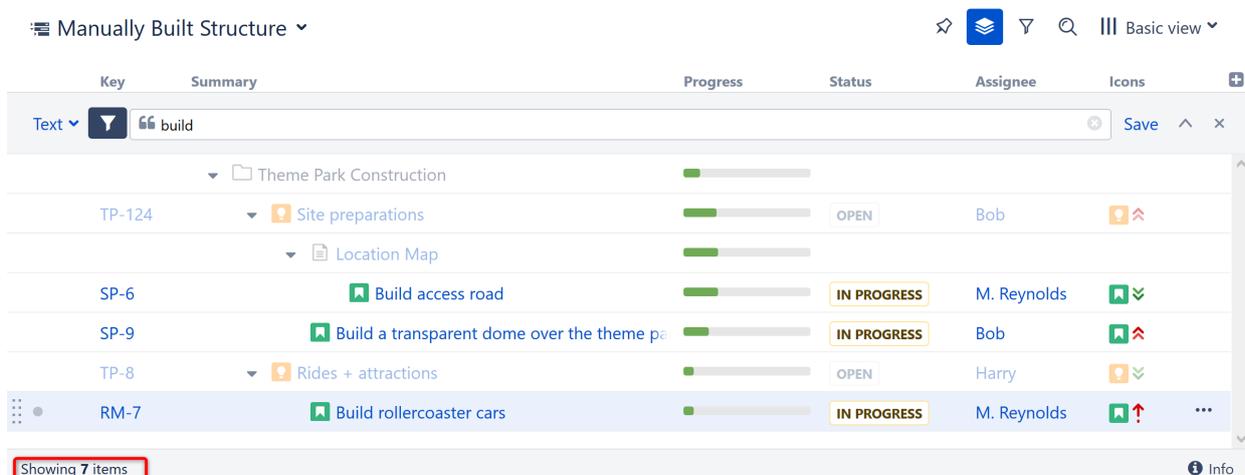
As with a JQL search, the checkmark  or exclamation  indicators will let you know whether the query is valid or not.

### 2.4.2 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.

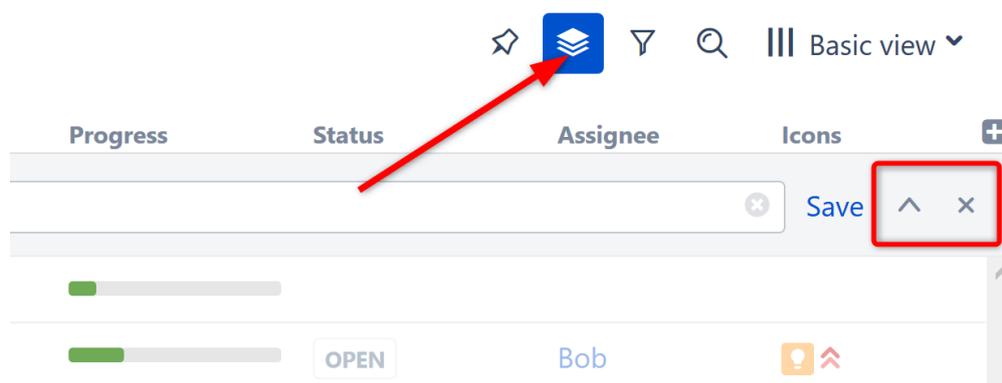


In the status bar at the bottom, you will see an updated items count.

✔ You can apply the filter to any [text](#), [JQL](#) or [S-JQL search](#).

## Filter Is a Transformation

When you apply the filter, you are also creating a [Transformation](#). The Transformations button is highlighted, showing that a transformation has been applied.



## 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.

i Filtering mode remains active even if you navigate to another page.

## 2.4.3 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](#) out all but those issues assigned to the current user or to sort issues by progress (or both!).

## Transformations vs. Generators

Transformations use the same types of functions as [generators](#). 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.

### 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.

### 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](#).

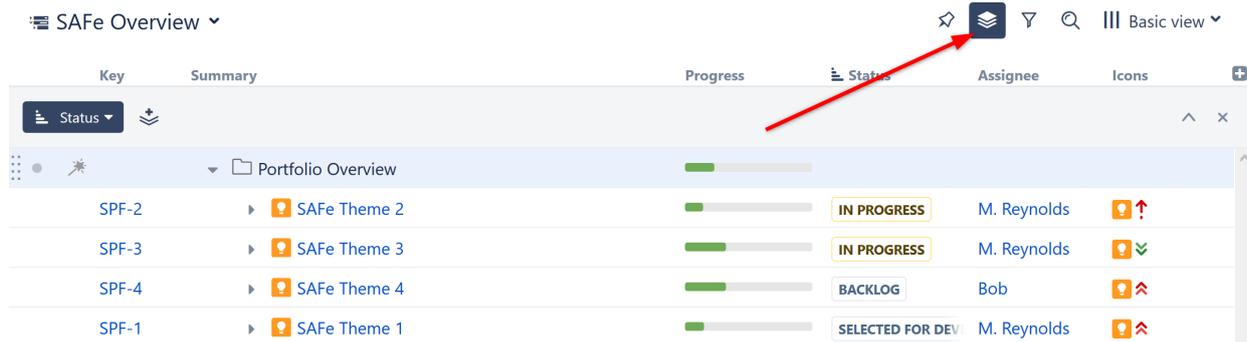


The Filter by Sprint transformation only affects folders, not issues, and it is applied to the whole structure. The Filter by Sprint generator is only applied to embedded sub-structures.

### Working with Transformations

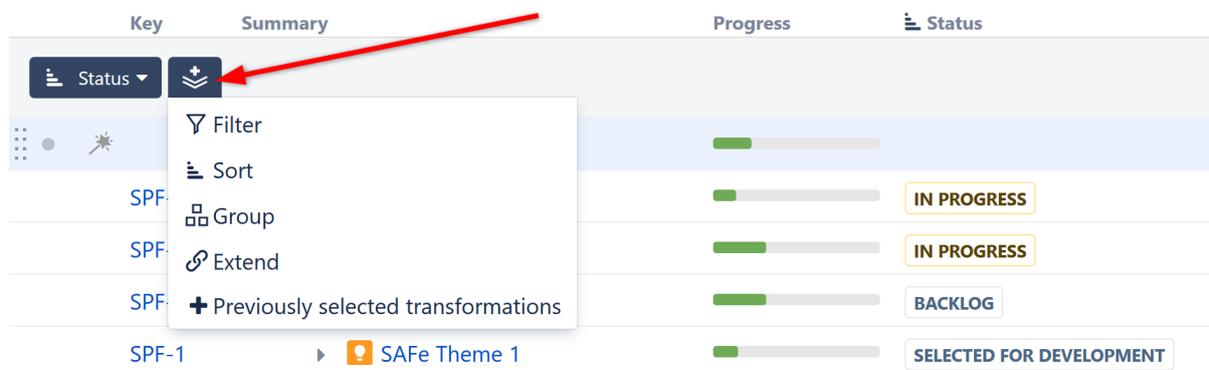
#### Using Transformations

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.



## Create a New Transformation

To add a new transformation, click the Add Transformation button and select one of the available transformations.



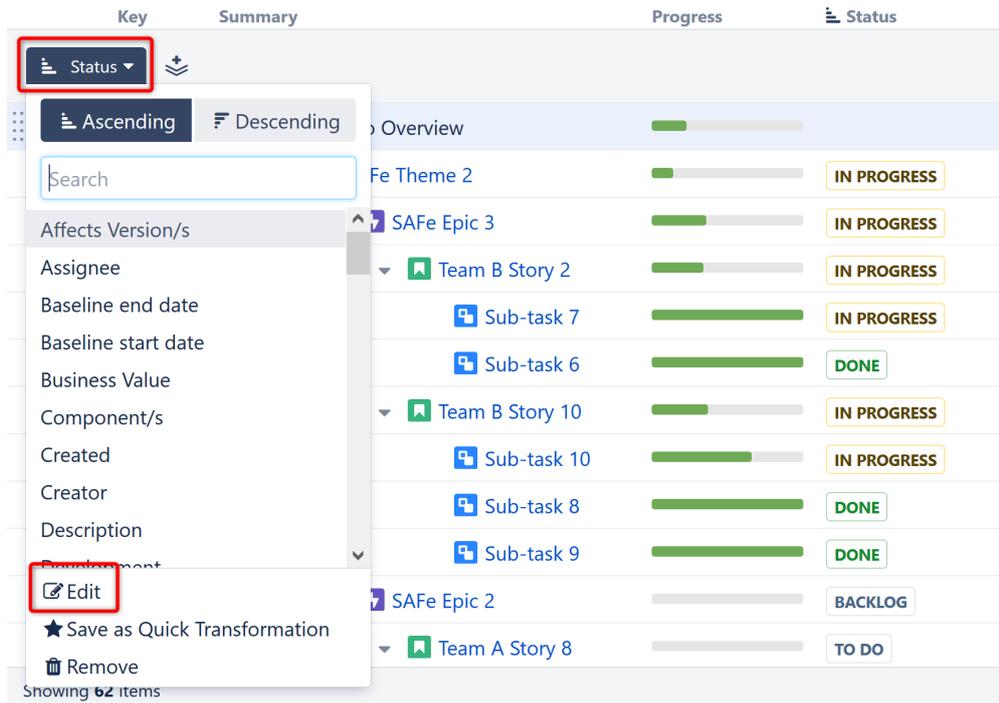
Transformations function in much the same way as their corresponding generators. For more information about how each works, see the documentation on [Types of Generators](#).

**i** 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**.

## 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.



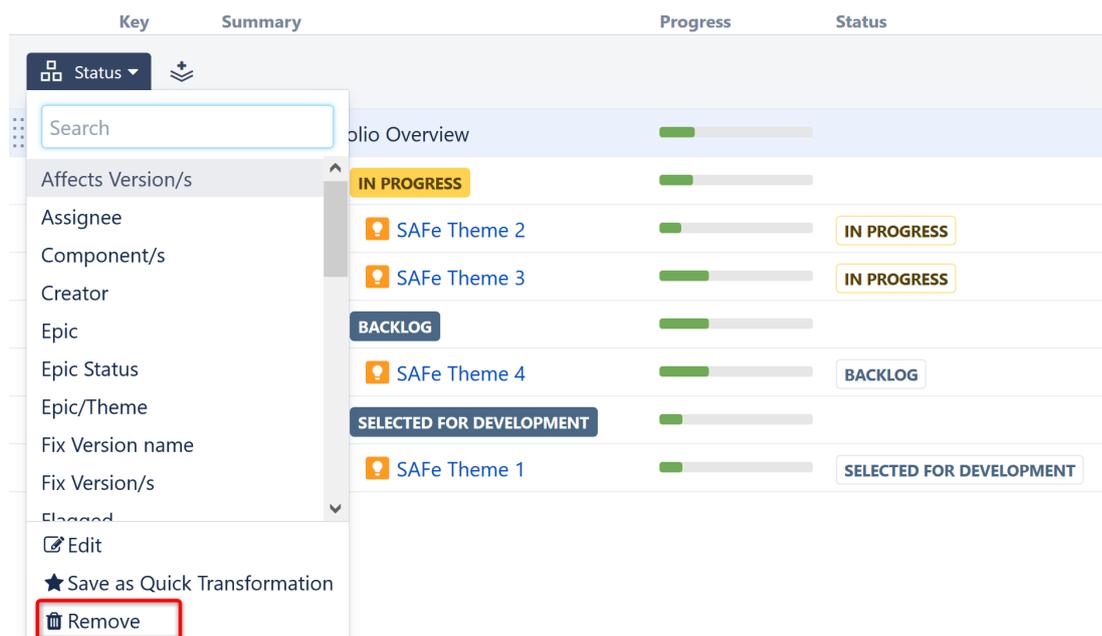
For more information about the advanced options available for each transformation, see the documentation for the corresponding [generator](#).



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](#) to learn more.

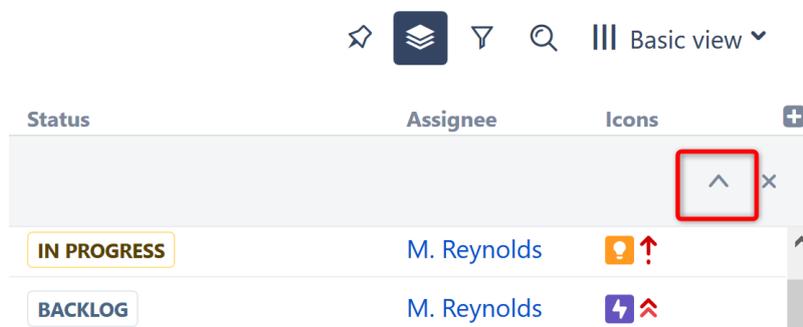
## Remove a Transformation

To remove a transformation, click the transformation name and select **Remove** from the bottom of the drop-down menu.



## 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](#). Click the transformation you want to save, and select **Save as Quick Transformation** at the bottom of the drop-down menu.

Key	Summary	Progress	Status
blocks	SAFe Theme 2	<div style="width: 50%;"><div style="background-color: green; height: 5px;"></div></div>	IN PROGRESS
	SAFe Epic 2	<div style="width: 10%;"><div style="background-color: green; height: 5px;"></div></div>	BACKLOG
	Team A Story 8	<div style="width: 0%;"><div style="background-color: green; height: 5px;"></div></div>	TO DO
	Sub-task 1	<div style="width: 0%;"><div style="background-color: green; height: 5px;"></div></div>	IN PROGRESS
	Sub-task 2	<div style="width: 0%;"><div style="background-color: green; height: 5px;"></div></div>	IN PROGRESS
	Team A Story 9	<div style="width: 10%;"><div style="background-color: green; height: 5px;"></div></div>	TO DO
	Sub-task 4	<div style="width: 0%;"><div style="background-color: green; height: 5px;"></div></div>	TO DO
	Team A Story 3	<div style="width: 20%;"><div style="background-color: green; height: 5px;"></div></div>	IN PROGRESS
	Team A Story 10	<div style="width: 0%;"><div style="background-color: green; height: 5px;"></div></div>	TO DO
	Team A Story 13	<div style="width: 0%;"><div style="background-color: green; height: 5px;"></div></div>	TO DO



You must have Control permission for a structure to save Quick Transformations.

## 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](#).
- To remove the sorting, click the **Summary** column header.

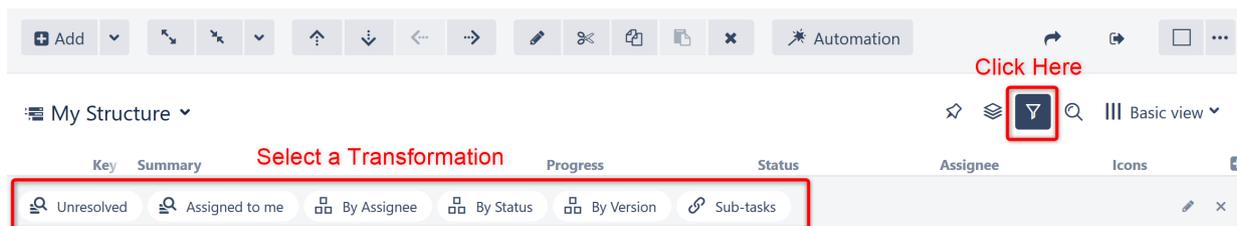
To apply a **filter**, you can run a [search](#) and then [filter](#) out non-matching items.

## Quick Transformations

Quick Transformations (sometimes called "Quick Filters") allow you to apply commonly-used transformations with the click of a button.

### 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.

**i** The Quick Transformation panel shows transformations that are associated with the current structure. See [Defining Quick Transformations](#) 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.

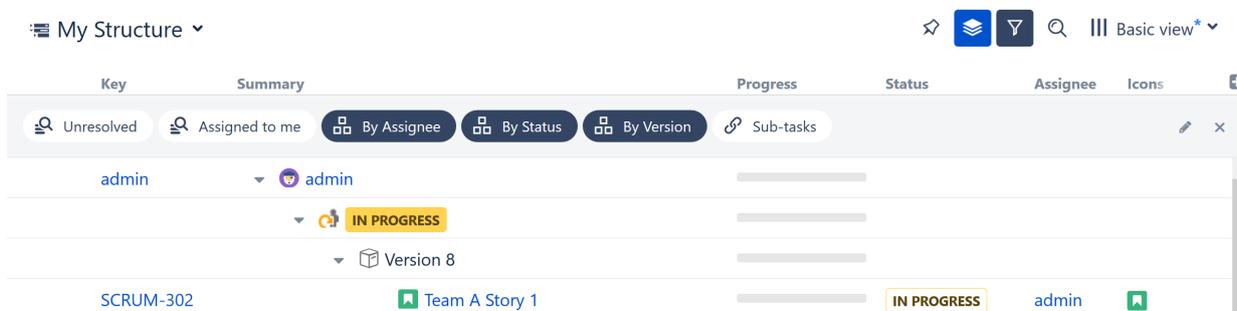
To learn more, see [Default Quick Transformations](#).

## 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:

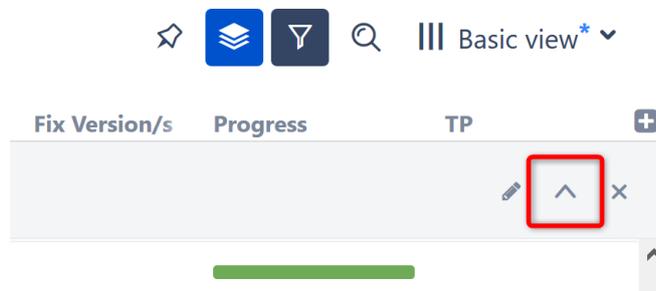
1. Group the first level by Assignee
2. Group the second level by Status
3. Group the third level by Version



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.

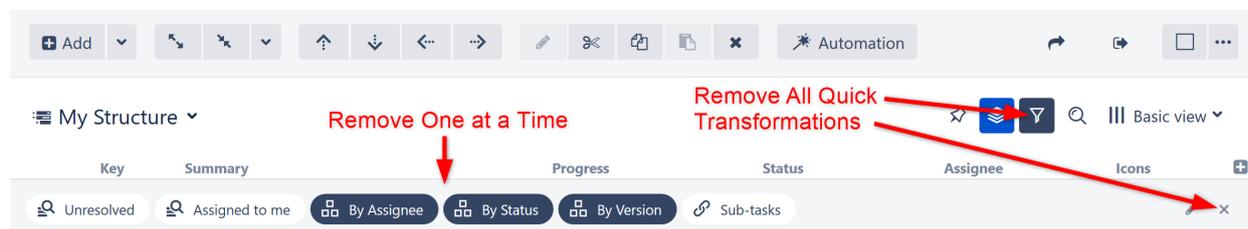


When the panel is hidden but quick transformations are applied, the Quick Transformations button will become blue .

## 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.

## 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](#)

2. Create your transformation
3. Click **Save as Quick Transformation**

The screenshot shows a Jira board with columns: Key, Summary, Progress, Status, Assignee, and Icons. A context menu is open over an Epic item. The menu items are: Edit (with a red circle 3), Save as Quick Transformation (highlighted with a red box), Remove, Affects Version/s, Assignee, Component/s, Creator, Epic, Epic Status, Epic/Theme, Fix Version name, and Fix Version/s. The board shows various items with progress bars and status labels like 'BACKLOG', 'TO DO', and 'SELECTED FOR DEVELOPMENT'.



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](#)
2. Enter your query (using Text, JQL, or S-JQL search)
3. Click **Save**

The screenshot shows a Jira board with columns: Key, Summary, Progress, Status, Assignee, and Icons. A search bar is open at the top, containing the JQL query 'issuetype = Bug'. The search bar is highlighted with a red box. The 'Save' button is also highlighted with a red box. The board shows various items with progress bars and status labels like 'BACKLOG', 'TO DO', and 'SELECTED FOR DEVELOPMENT'.

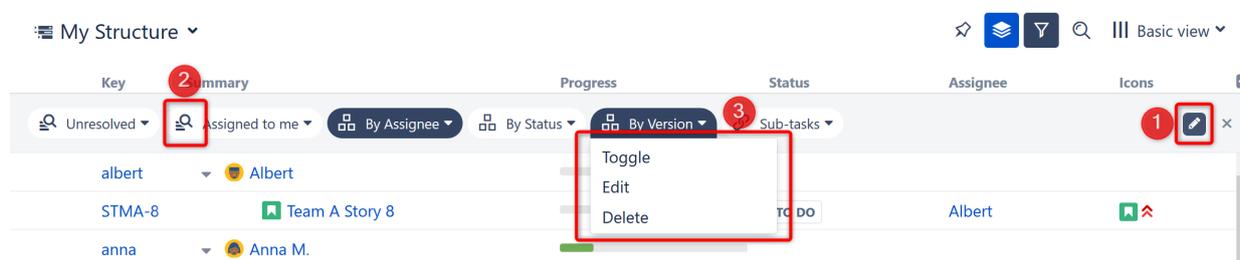
After clicking **Save**, you will be given the opportunity to name and configure the quick transformation.

## 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. Click the edit icon on the right side of the Quick Transformations panel
2. To **move a transformation**, drag its icon to the desired position
3. To **toggle, edit 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.

## 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.

### 2.4.4 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.

☰ Manually Built Structure ▾     ||| Compact ▾

Key	Summary	
	▾  Theme Park Construction	
TP-124	▾  Site preparations	
SP-9	▾  Build a transparent dome over the theme	
 ● QA-7	 Check seismic activity	
SP-6	▾  Build access road - We need an eight-la	
 QA-7	 Check seismic activity	



Structure Panel on the [Issue Page](#) automatically pins the issue being displayed, so you only see the relevant part(s) of the structure.

### 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.

## Turning Pinned Item Mode On and Off

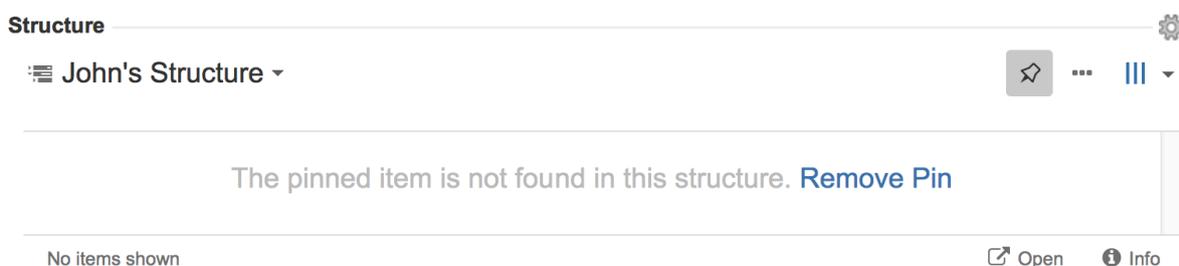
To turn Pinned Item Mode on or off, click the Pin button on the panel toolbar or press **Ctrl+.** on your keyboard.

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 is 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.

## 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.

## 2.4.5 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.

### Finding Duplicates

If any items appear more than once in a structure, a duplicates counter will appear in the status bar.

Showing 8 items x2 2 items with duplicates

- ⓘ 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.

Key	Summary	Σ Story	Σ Time	Progress	TP	Assignee	Status	WSJF (Basic)	Σ Story	Story P
STMB-3	Story 1				🟢	Unassign	TO DO	800		
STMB-3	Task A				🟢	Unassign	TO DO	800		
STMB-3	Big Bug				🔴	Unassign	TO DO	800		
STMB-3	Task B				🟢	Unassign	TO DO	800		
STMB-3	Story 2				🟢	Unassign	TO DO	800		
STMB-3	Task A				🟢	Unassign	TO DO	800		
STMB-3	Big Bug				🔴	Unassign	TO DO	800		
STMB-3	Story 3				🟢	Unassign	TO DO	800		

5 rows with duplicates Filter Pin ↓ ↑  Mark duplicate rows

Showing 8 items x2 2 items with duplicates Info

### 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

## 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.

Key	Summary	Σ Story	Σ Time	Progress	TP	Assigne	Status	WSJF (Basic)	Σ Story	Story Po
STMB-	Story 1					Unassign	TO DO	800		
 STMB-	Task A					Unassign	TO DO	800		
 STMB-	Big Bug					Unassign	TO DO	800		
STMB-	Task B					Unassign	TO DO	800		
STMB-	Story 2					Unassign	TO DO	800		
 STMB-	Task A					Unassign	TO DO	800		
 STMB-	Big Bug					Unassign	TO DO	800		
STMB-	Story 3					Unassign	TO DO	800		

5 rows with duplicates  Filter  Pin  Mark duplicate rows

Showing 8 items  2 items with duplicates Info



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.

Key	Summary	Σ Story	Σ Time	Progress	TP	Assigne	Status	WSJF (Basic)	Σ Story	Story Po
STMB-	Story 1				▲	Unassig	TO DO	800		
STMB-	Task A				▲	Unassig	TO DO	800		
STMB-	Big Bug				▲	Unassig	TO DO	800		
STMB-	Task B				▲	Unassig	TO DO	800		
STMB-	Story 2				▲	Unassig	TO DO	800		
STMB-	Task A				▲	Unassig	TO DO	800		
STMB-	Big Bug				▲	Unassig	TO DO	800		
STMB-	Story 3				▲	Unassig	TO DO	800		

5 rows with duplicates Filter Pin ↓ ↑  Mark duplicate rows ×

Showing 8 items 2 items with duplicates Info

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.

## Filtering by Duplicates

Clicking the Filter button hides everything in the structure except duplicate items and their parents.

Key	Summary	Σ Story	Σ Time	Progress	TP	Assigne	Status	WSJF (Basic)	Σ Story	Story Po
STMB-	Story 1				▲	Unassig	TO DO	800		
STMB-	Task A				▲	Unassig	TO DO	800		
STMB-	Big Bug				▲	Unassig	TO DO	800		
STMB-	Story 2				▲	Unassig	TO DO	800		
STMB-	Task A				▲	Unassig	TO DO	800		
STMB-	Big Bug				▲	Unassig	TO DO	800		
STMB-	Story 3				▲	Unassig	TO DO	800		
STMB-	Big Bug				▲	Unassig	TO DO	800		

5 rows with duplicates Filter Pin ↓ ↑  Mark duplicate rows ×

Showing 5 items 2 items with duplicates Info



When the Duplicates panel is closed, the Duplicates filter is disabled automatically.

## 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.

Key	Summary	Σ Story	Σ Time	Progress	TP	Assignee	Status	WSJF (Basic)	Σ Story	Story Pct
STMB-3	Story 1					Unassign	TO DO	800		
STMB-3	Task A					Unassign	TO DO	800		
STMB-3	Big Bug					Unassign	TO DO	800		
STMB-3	Story 2					Unassign	TO DO	800		
STMB-3	Task A					Unassign	TO DO	800		
STMB-3	Big Bug					Unassign	TO DO	800		

4 rows with duplicates Filter  Pin ↓ ↑  Mark duplicate rows ×

Showing 4 items  2 items with duplicates Info



You can quickly pin items by simply clicking the 'x2' icon beside any duplicate item.

## 2.5 Formulas

Formulas can serve a variety of purposes within a structure, including:

- Performing simple or complex calculations based on issue fields or other attributes
- Comparing values from multiple fields
- Creating a visual notification based on other fields or calculations
- Adding [wiki markup](#)

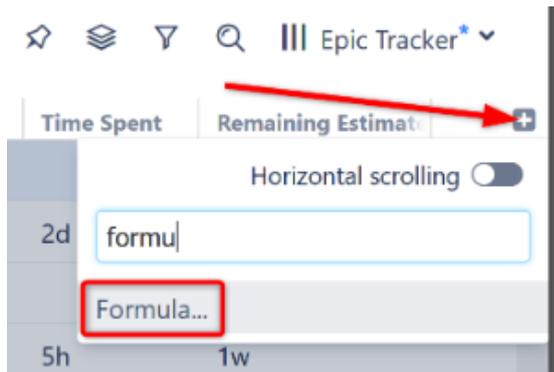
The following articles will show you how to create your own custom formulas or add one of our [predefined formulas](#) to a structure.

### 2.5.1 Formula Columns

The following article will walk you step-by-step through the creation of a basic [formula](#).

#### Add a Formula Column

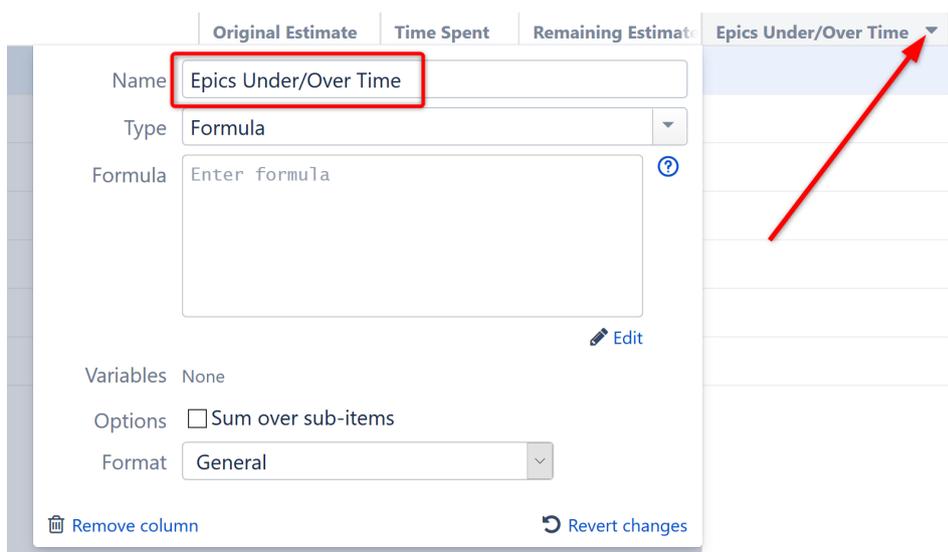
Start with [adding a new column](#) by clicking the + icon to the right of the column headings. Select **Formula...** as its type.



## Name Your Formula

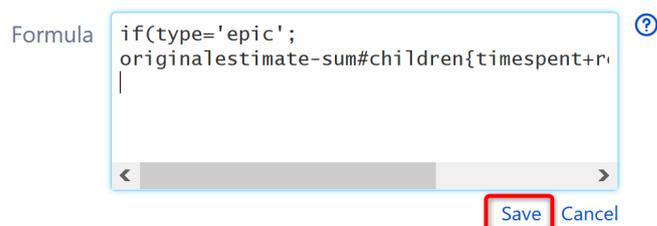
Open the Formula editor by clicking the downward-pointing triangle next to the new column header.

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 click **Save**.



Formulas should be constructed using the [Expr Language](#), an intuitive language that supports variables, arithmetic operations and functions. You can find tutorials, reference guides, and several examples in our [Expr Language documentation](#).



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](#), 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:  
`timespent+remainingestimate`
3. Total that value for the issue and all its children: `sum#children`
4. Subtract that total from the epic's Original Estimate: `originalestimate-sum#children{timespent+remainingestimate}`

## 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. The results of your formula will also appear in the new column.

	Original Estimate	Time Spent	Remaining Estimate	Epic Under/Over Time
Name	Epic Under/Over Time			255600000
Type	Formula			
Formula	<pre>if(type='epic'; originalestimate-sum#children{timespent+re</pre>			
Variables	<ul style="list-style-type: none"> <li>✓ type</li> <li>✓ originalestimate</li> <li>✓ timespent</li> <li>✓ remainingestimate</li> </ul> <p>4 variables used. Click a variable to define it.</p>			
Options	<input type="checkbox"/> Sum over sub-items			
Format	General			
				-100800000

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.

## Handling Errors

Formula errors are typically due to one of the following:

- **Syntax Error** - occurs when the formula cannot be parsed. Chances are, you missed a closing parenthesis or other punctuation. Review the red highlighted sections and consult our [Expr Language](#) guide if you're unsure how to correct it.
- **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](#) to ensure you're using a supported function.
- **Variable Resolution Error** - occurs when you have used a variable that hasn't been defined yet. You don't have to correct these errors now. We'll address them when we [define variables](#).

## 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.



The results above aren't very useful in their current format, because our results are being displayed in milliseconds! We'll change them to the more useful Duration format below.

## Variables

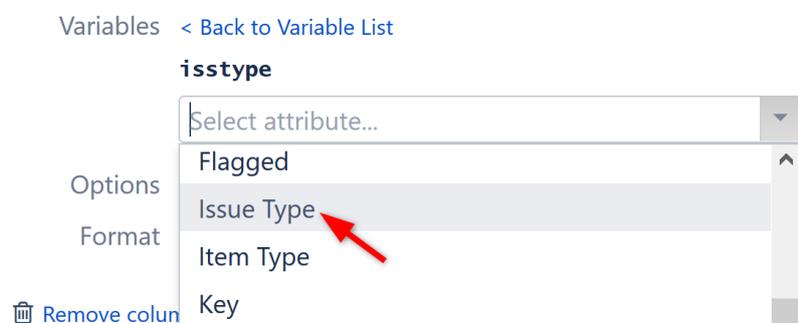
Most formulas will contain at least one variable (otherwise, the result will be the same for each row in the structure). These variables need to be mapped to *attributes*, which can be issue fields, progress, a hierarchical total, user properties, [another column](#) or even [another formula](#).

As you write your formula, Structure attempts to map your variables to well-known attributes. If Structure is unable to map your variables, you will receive an error and need to map the variable manually.

Variables ! `isstype`  
✓ `originalestimate`  
✓ `timespent`  
✓ `remainingestimate`

4 variables used. Click a variable to define it.

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.



i The following names are automatically recognized by Structure:

- Names of standard Jira fields, such as **Summary** or **Priority**.
- Names of custom fields, with all non-letters removed and all spaces converted to underscores. For example, **Story\_Points**.
- Names starting with **Total\_** or **Sum\_** and having a well-known name afterwards, such as **Sum\_Story\_Points** or **Total\_Estimate**. These are converted to a [Sum](#) attribute of the given value (without the duplicate removal option).

 Even if Structure successfully maps your variables, it's still a good idea to review them!

To learn more about assigning variables within a formula, see [Columns as Variables](#).

## 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.

Options  Sum over sub-items  
 Exclude duplicates  
 After filtering

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 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.

 You can also use [aggregate functions](#) 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](#) for more information.
- **Wiki Markup** - allows you to add wiki markup, including colors and images, to a column. See [Wiki Markup in Formula Columns](#) for more details.

Going back to our sample formula, let's change the Format to **Duration** and check **Work time**.

Format   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:

Simple Structure ▼ ☆ ☰ 🔍 ||| Epic Tracker\* ▼

Key	Summary	Original Estimate	Time Spent	Remaining Estimate	Epic Under/Over Time
STMB-40	<span>+</span> Epic 1	5w			1w 3d 7h
STMB-31	<span>+</span> Story 1		2d	3d 4h	
STMB-32	<span>+</span> Story 2	1w		1w	
STMB-42	<span>+</span> Story 3		5h	1w	
STMB-41	<span>+</span> Epic 2	5w		4w	-3d 4h
STMB-37	<span>+</span> Story 4	2w	1w 2d	3d	
STMB-39	<span>+</span> Story 6	2w	3w 1d	2d 4h	



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 are treated just like any other column, so they can be shared by:

- Making them a part of a public or shared [View](#), which other users can select
- Creating a [perspective URL](#) that will open the structure with the same configuration, including the formula column

### Sorting by Calculated Value

You can [sort](#) by the values calculated in a formula column by clicking the column header.

### See Also

- [Creating an Advanced Formula Column](#)
- [Bundled Formulas](#)
- [Expr Language](#)

### Columns as Variables

When using a [formula](#), 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 screenshot displays the configuration for a formula column named "Epic Time Warning". The formula is "If (epicstime < 0; \"WARNING!\")". The "Used in Columns" section is highlighted with a red box, showing a dropdown menu with options: "Epic Under/Over Time (Formula)", "Original Estimate", and "Remaining Estimate".

## Data is Copied, Not Linked

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. 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!](#))

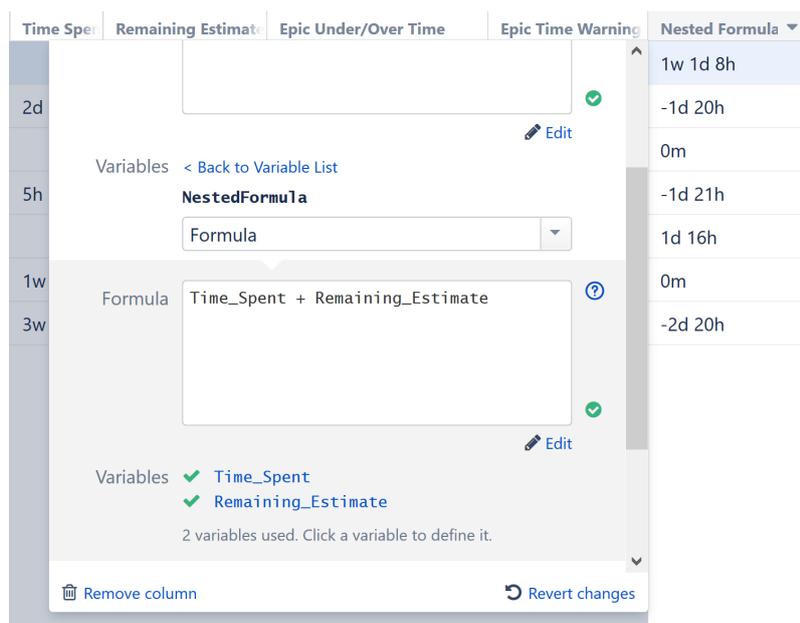
Key	Summary	Original Estimate	Time Spent	Remaining Estimate	Epic Under/Over Time	Epic Time Warning
STMB-40	Epic 1	5w			1w 3d 7h	
STMB-31	Story 1		2d	3d 4h		
STMB-32	Story 2	1w		1w		
STMB-42	Story 3		5h	1w		
STMB-41	Epic 2	5w		4w	-3d 4h	WARNING!
STMB-37	Story 4	2w	1w 2d	3d		
STMB-39	Story 6	2w	3w 1d	2d 4h		

Once we've created this new formula, we could delete our original Epic Under/Over Time column (if we wanted to). The Epic Time Warning column 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 no longer exists.

## Formulas as Variables

When using a [formula](#), you can assign another formula as a variable. This is similar to using [columns as variables](#), except that the formula doesn't have to be in a column already - you can enter the formula exactly as you want it to work for the new variable.

To create a nested formula, simply select **Formula...** in the attribute selection drop-down. A new Formula field will appear for the variable.



Once you finish setting up a nested formula, you can collapse the dialog by clicking **< Back to Variable List**. To edit the nested formula later, simply select the variable from the Variables list.

## Variables in Nested Formulas

Nested formulas can have their own variables.

Variables in a nested formula are not the same as the variables declared by a parent formula. Variables do not overwrite each other, even if they have the same name.

## Unlimited Nesting Levels

Nested formulas can also use formulas as variables. Doing this, you can create very complicated formulas that rely on several levels of nested formulas.

There is no nesting level limit.



A word of caution - the more nested formulas you include, the more difficult it becomes to troubleshoot the column.

## Query Match as Variables

Query match allows you to limit your formula to only those items that match a specified text, JQL or [S-JQL](#) query.

The screenshot shows a Jira table with columns: Key, Summary, Status, Progress, Assignee, Σ Creator = Admin, and Formula. The table contains several rows of tasks, including 'Theme Park is Safe' campaign, 'Team A Story 1', 'SAFe Epic 2', and various 'Team A Story' and 'Sub-task' items. A modal window for editing a formula is open, showing the formula 'If(queryresult, "A-Team")' and a query 'Team A'.

### Query Match as a Variable

To use Query match in a formula:

1. Assign a variable for your query.
2. In the Variables section, assign that variable to Query match.
3. Select the Query type. You can use Text, JQL or [S-JQL](#) queries.
4. Enter the query.

**i** If you've already created a [Query Match column](#), you can also [use the existing column as a variable](#).

## Work Time in Formula Columns

When the format of a [formula column](#) 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.

**i** 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.

## 2.5.2 Wiki Markup in Formula Columns

Customize your structure, call attention to critical information or color-code data fields using wiki markup within [formula columns](#).

Wiki markup allows you to:

- Specify the text color within a column
- Highlight cells with background coloring
- Insert images
- Add emojis

## Using Wiki Markup

To add wiki markup to a formula column:

Click the Add Column button (+) and select **Formula**

1. Enter a column name.
2. Include wiki markup language in your formula column, surrounded by double quotes ("). See [Markup Options](#) below for more details.
3. Under the Format menu, select Wiki Markup. (This is important - your content will not display correctly unless the Wiki Markup format is selected.)

Name

Type

Formula  ?

✔ [Edit](#)

Variables ✔ [issuetype](#)  
One variable used. Click the variable to define it.

Options  Sum over sub-items

Format

[Remove column](#)

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:*

SAFe Structure ▾
☆ ⚙️ ▾ 🔍 ||| Basic view\* ▾

Key	Summary	Σ Story Points	Assignee	Pr	TP	Epic Markup
SPR-6	▶️ SAFe Epic 6	46	Bob	⚪	⬆️ ⬆️ ⬆️	Epic
SPR-5	▾️ SAFe Epic 5	42	M. Reynolds	⚪	⬆️ ⬆️ ⬆️	Epic
STMA-16	Team A Story 16	15	Unassigned	⚪	⬆️ ⬆️	
STMA-4	Team A Story 4	15	C. Bacca (Inactive)	⚪	⬆️ ⬆️	
STMA-5	Team A Story 5	12	C. Bacca (Inactive)	⚪	⬆️ ⬆️	
SPR-4	▾️ SAFe Epic 4	22	M. Reynolds	⚪	⬆️ ⬆️ ⬆️	Epic
STMA-7	Team A Story 7	9	Jack Brown	⚪	⬆️ ⬆️	
STMA-14	Team A Story 14	13	Unassigned	⚪	⬆️ ⬆️	
SPR-3	▶️ SAFe Epic 3	48	Bob	⚪	⬆️ ⬆️ ⬆️	Epic
SPR-2	▾️ SAFe Epic 2	51	M. Reynolds	⚪	⬆️ ⬆️ ⬆️	Epic

Showing 67 items 3 items with duplicates Info

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](#).



All markup language should be included between double quotes (").



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.



It's fine to mix text and emojis/images in the same column, just not the same cell.



Name

Type

Formula

Variables  DueDate  
One variable used. Click the variable to define it.

Options  Sum over sub-items

Format

Remove column



We used text to call attention to overdue items, but you could also add a flag: "(flag)"

To learn more about using *If statements*, *DAY\_BETWEEN*, or any other functions, see [Expr Function Reference](#).

## 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.

SAFe Structure ▾ ☆ ☰ ▾ 🔍 ||| Basic view\* ▾

Key	Summary	Project Symbol	Project	Assignee	Pr
✓ STMB-22	Sub-task 8	★	SAFe Team B	Mary	▬
✓ STMB-19	Sub-task 6	★	SAFe Team B	Nah Duo	▬
MKT-4	'Theme Park is Safe' campaign	★	Marketing	Demo User	▬
MKT-2	Celebrity endorsements	★	Marketing	C. Bacca (Inactive)	▬
MKT-1	Anti-PR campaign to discredit safety of competing the	★	Marketing	Man in Black	▬
STMA-1	Team A Story 1	★	SAFe Team A	C. Bacca (Inactive)	▬
SPR-12	SAFe Epic 12	★	SAFe Program	Unassigned	▬
SPR-11	SAFe Epic 11	★	SAFe Program	M. Reynolds	▬
SPR-10	SAFe Epic 10	★	SAFe Program	Unassigned	▬
SPR-9	SAFe Epic 9	★	SAFe Program	Bob	▬

Showing 67 items 3 items with duplicates Info

To create this column, we used the special character notations for stars "(\*)" - along with color designations:

If (Project = "SAFe Program"; "(\*b)"; Project = "SAFe Team A"; "(\*y)"; Project = "SAFe Team B"; "(\*r)"; Project = "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!

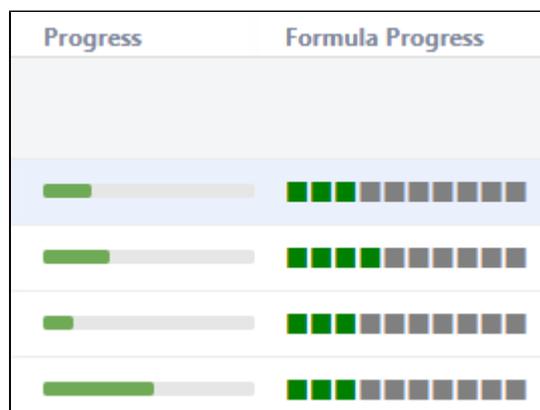
## Wiki Markup Advanced Examples

While wiki markup has its limitations, you can get pretty creative and visualize more complex metrics in Structure columns, such as custom progress bars, bar charts and much more. The use of colors and images greatly expands the possibilities.

In this article, we've put together several advanced, customizable examples of wiki markup usage:

### 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 customProgress=<formulaForProgress>: CONCAT("{color:green}",
REPEAT(" ", FLOOR(customProgress/10)), "{color}{color:gray}",REPEAT
(" ", 10-FLOOR(customProgress/10)), "{color}")
```

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 customProgress=SUM{progressField}/SUM{1}: CONCAT("{color:
green}", REPEAT(" ", FLOOR(customProgress/10)), "{color}{color:
gray}", REPEAT(" ", 10-FLOOR(customProgress/10)), "{color}")
```

**i** 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

Multi-bar	Multi-bar different character	Multi-bar with image	Multi-bar with numbers

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

```
//Granularity - length of the bar in characters; bar - filler
character for the bar

with granularity=20: with bar = " ":

//Lengths of bar sections per criterion in characters

with todo=FLOOR(COUNT#truthy{status="to do"}/COUNT{1}
*granularity): with inprogress=FLOOR(COUNT#truthy{status="in
progress"}/COUNT{1}*granularity): with done=FLOOR(COUNT#truthy
{status="done"}/COUNT{1}*granularity): with other=granularity-
todo-inprogress-done:

//Bar chart

CONCAT("{color:red}",REPEAT(bar, todo), "{color}{color:orange}",
REPEAT(bar, inprogress),"{color}{color:green}", REPEAT(bar, done),
"{color}{color:gray}", REPEAT(bar, other), "{color}")
```

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 granularity=200:

//Lengths of bar sections per criterion

with todo=FLOOR(COUNT#truthy{status="to do"}/COUNT{1}
*granularity): with inprogress=FLOOR(COUNT#truthy{status="in
progress"}/COUNT{1}*granularity): with done=FLOOR(COUNT#truthy
{status="done"}/COUNT{1}*granularity): with other=granularity-
done-inprogress-todo:

//Bar chart using simple square graphics

CONCAT("!https://www.example.com/images/Red.png|height=20,width=",
todo,"!", "!https://www.example.com/images/Orange.png|height=20,
width=", inprogress, "!", "!https://www.example.com/images/Green.
png|height=20,width=", done, "!", "!https://www.example.com/images
/Gray.png|height=20,width=", other, "!")

```

## 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 granularity=20: with bar="":

//Tracked criteria: valueN - actual number of issues with that
criterion; value - length of the criterion bar in characters

with all=COUNT{1}: with todoN=COUNT#truthy{status="to do"}: with
todo=FLOOR(todoN/all*granularity): with inprogressN=COUNT#truthy
{status="in progress"}: with inprogress=FLOOR(inprogressN
/all*granularity): with doneN=COUNT#truthy{status="done"}: with
done=FLOOR(doneN/all*granularity): with otherN=all-todoN-
inprogressN-doneN: with other=granularity-todo-inprogress-done:

//Bar chart. If the number of symbols in the bar is longer by 2
characters than the length of the number of issues with the
criterion, the latter number is displayed in the middle of the
bar, replacing a corresponding number of filler characters

```

```

CONCAT( "{color:red}", IF(todo>=LEN(todoN)+2, CONCAT(REPEAT(bar,
FLOOR((todo-LEN(todoN))/2)), todoN, REPEAT(bar, FLOOR((todo-LEN
(todoN))/2)+MOD(todo-LEN(todoN), 2))), REPEAT(bar, todo)),
"{color}{color:orange}", IF(inprogress>=LEN(inprogressN)+2, CONCAT
(REPEAT(bar, FLOOR((inprogress-LEN(inprogressN))/2)), inprogressN,
REPEAT(bar, FLOOR((inprogress-LEN(inprogressN))/2)+MOD(inprogress-
LEN(inprogressN), 2))), REPEAT(bar, inprogress)),
"{color}{color:green}", IF(done>=LEN(doneN)+2, CONCAT(REPEAT(bar,
FLOOR((done-LEN(doneN))/2)), doneN, REPEAT(bar, FLOOR((done-LEN
(doneN))/2)+MOD(done-LEN(doneN), 2))), REPEAT(bar, done)),
"{color}{color:gray}", IF(other>=LEN(otherN)+2, CONCAT(REPEAT(bar,
FLOOR((other-LEN(otherN))/2)), otherN, REPEAT(bar, FLOOR((other-LEN
(otherN))/2)+MOD(other-LEN(otherN), 2))), REPEAT(bar, other)), "{col
or}" )

```

## 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 day1_created=COUNT#truthy{DATE_SUBTRACT(NOW(), 6, "days")
<=created and DATE_SUBTRACT(NOW(), 5, "days")>created}:
with day1_resolved=COUNT#truthy{DATE_SUBTRACT(NOW(), 6, "days")
<=resolved and DATE_SUBTRACT(NOW(), 5, "days")>resolved}:
with day2_created=COUNT#truthy{DATE_SUBTRACT(NOW(), 5, "days")
<=created and DATE_SUBTRACT(NOW(), 4, "days")>created}:

```

```

with day2_resolved=COUNT#truthy{DATE_SUBTRACT(NOW(),5,"days")
<=resolved and DATE_SUBTRACT(NOW(),4,"days")>resolved}:
with day3_created=COUNT#truthy{DATE_SUBTRACT(NOW(),4,"days")
<=created and DATE_SUBTRACT(NOW(),3,"days")>created}:
with day3_resolved=COUNT#truthy{DATE_SUBTRACT(NOW(),4,"days")
<=resolved and DATE_SUBTRACT(NOW(),3,"days")>resolved}:
with day4_created=COUNT#truthy{DATE_SUBTRACT(NOW(),3,"days")
<=created and DATE_SUBTRACT(NOW(),2,"days")>created}:
with day4_resolved=COUNT#truthy{DATE_SUBTRACT(NOW(),3,"days")
<=resolved and DATE_SUBTRACT(NOW(),2,"days")>resolved}:
with day5_created=COUNT#truthy{DATE_SUBTRACT(NOW(),2,"days")
<=created and DATE_SUBTRACT(NOW(),1,"days")>created}:
with day5_resolved=COUNT#truthy{DATE_SUBTRACT(NOW(),2,"days")
<=resolved and DATE_SUBTRACT(NOW(),1,"days")>resolved}:
with day6_created=COUNT#truthy{DATE_SUBTRACT(NOW(),1,"days")
<=created and DATE_SUBTRACT(NOW(),8,"hours")>created}:
with day6_resolved=COUNT#truthy{DATE_SUBTRACT(NOW(),1,"days")
<=resolved and DATE_SUBTRACT(NOW(),8,"hours")>resolved}:
with day7_created=COUNT#truthy{DATE_SUBTRACT(NOW(),8,"hours")
<=created}:
with day7_resolved=COUNT#truthy{DATE_SUBTRACT(NOW(),8,"hours")
<=resolved}:
with maxth=MAX(day1_created, day1_resolved, day2_created,
day2_resolved, day3_created, day3_resolved, day4_created,
day4_resolved, day5_created, day5_resolved, day6_created,
day6_resolved, day7_created, day7_resolved):
//25 is maximum working height
with heighth=25:
IF(itemtype!="issue",CONCAT(
"!https://www.example.com/images/Red.png|height=", FLOOR
(day1_created/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Green.png|height=", FLOOR
(day1_resolved/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Red.png|height=", FLOOR
(day2_created/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Green.png|height=", FLOOR
(day2_resolved/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Red.png|height=", FLOOR
(day3_created/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Green.png|height=", FLOOR
(day3_resolved/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Red.png|height=", FLOOR
(day4_created/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Green.png|height=", FLOOR
(day4_resolved/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Red.png|height=", FLOOR
(day5_created/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Green.png|height=", FLOOR
(day5_resolved/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Red.png|height=", FLOOR
(day6_created/maxth*heighth), ",width=5!",

```

```

"!https://www.example.com/images/Green.png|height=", FLOOR
(day6_resolved/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Red.png|height=", FLOOR
(day7_created/maxth*heighth), ",width=5!",
"!https://www.example.com/images/Green.png|height=", FLOOR
(day7_resolved/maxth*heighth), ",width=5!"
))

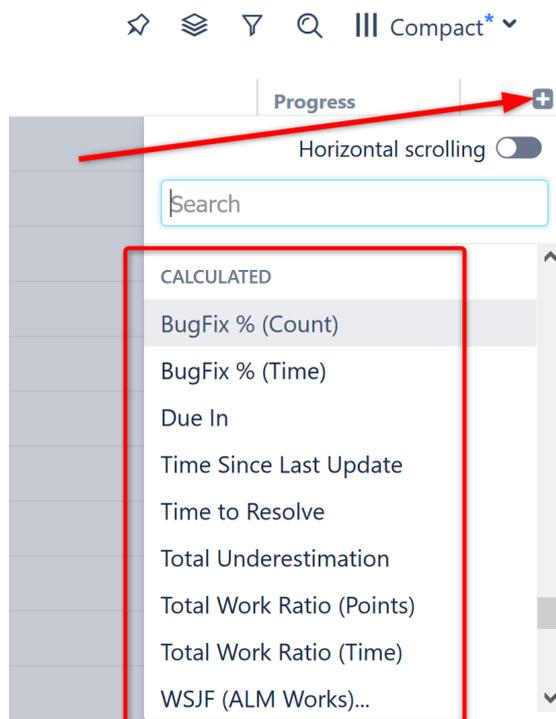
```

The criteria for issue inclusion can be easily customized to your team's needs. As mentioned above, we recommend hosting image files locally.

### 2.5.3 Bundled Formulas

The simplest way to include formulas in a structure is to use one of our bundled formulas.

To add a bundled formula, click the **+** button to the right of the column header and scroll down until you locate the **CALCULATED** section.



The following predefined formulas are available:

Column Name	Description
BugFix % (Count)	Displays the percentage of bugs among all sub-issues. Bugs are identified by having issue type "Bug".
BugFix % (Time)	

Column Name	Description
	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.
Time Since Last Update	Displays the amount of calendar time that has passed since the issue was last updated.
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 <a href="#">SUM function</a> 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 <a href="#">SUM function</a> 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 <a href="#">SUM function</a> 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.
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: <ul style="list-style-type: none"> <li>• Job Size</li> <li>• User/Business Value</li> </ul>

Column Name	Description
	<ul style="list-style-type: none"> <li>• Time Criticality</li> <li>• Risk Reduction</li> <li>• Opportunity Enablement</li> </ul> <p>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.</p>
WSJF (ALM Works)	<p>Weighted Shortest Job First metric, according to categories used at ALM Works:</p> <ul style="list-style-type: none"> <li>• Benefit</li> <li>• Pain</li> <li>• Marketability</li> <li>• Impact</li> <li>• Cost</li> <li>• Risk</li> <li>• Clarity</li> </ul> <p>To use this formula, you must set up such fields with the following values: Nil, Low, Medium and High.</p>
Assignee Cost	<p>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.</p>

### 2.5.4 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. When used in a [Formula Column](#), the expression is calculated for each visible row in the displayed structure or query result.

Expr is an easy language to learn, and yet it is powerful enough to create very complex formulas. The following guide will cover the basic requirements of the Expr language.

For a more in-depth study, see our [Expr Reference Guides](#).

- ✔ You can view examples of Expr formulas by adding [bundled formulas](#) to your structure. To see the formula, simply open the [column options](#) panel.

## Language Components

An expression may contain one or more of the following:

- Variables, which are mapped to *attributes*, including issue fields, progress, user properties, [another column](#) or even [another formula](#).
- Functions, which may take some arguments, and which produce the result at the moment of calculation.
- Numbers and text strings.
- Arithmetic, logical operations and parentheses.

There are also more advanced constructs:

- Aggregate Functions, which calculate some aggregate (like sum or average) of an expression's values calculated for multiple items in the structure.
- Local Variables, which let you introduce a value and reuse it multiple times in the formula.
- Comments, which allow you document larger formulas.

## Basic Constructs

### Variables

Variables are user-defined names, which represent *attributes*, such as:

- Issue fields
- Progress
- User properties
- [Item type](#)
- [Status category](#)
- [Another column](#), or
- [Another formula](#)

Variables can contain letters (English only), numbers, dot (".") or underscore ("\_") characters. Variables cannot contain spaces, and the first character must be a letter or an underscore.

Examples:

- `Priority`
- `remaining_estimate`
- `abc11`
- `sprint.name`

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. For this reason, it's best to choose meaningful names for your variables, rather than "x" or "VeryComplicatedCustomFieldName".

See [Formula Columns](#) for more information about creating and mapping variables.



Variable names are case-insensitive, meaning that `Priority`, `priority` and `pRiOrItY` will all refer to the same variable.

## Functions

A function calculates a value based on its arguments and, sometimes, some external aspect. A function 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()`

There are a number of standard functions available with Structure – see [Expr Function Reference](#) for details.

A function may take zero, one or more arguments. Some functions take variable number of arguments. Additionally, each argument can be another Expr expression and include calls to other functions.



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.



Function names are case-insensitive, like the variables. You can write `TODAY()` or `Today()`.

## Numbers and Text Strings

### Numbers

You can use numbers in your formula. Formulas support whole numbers, decimals, or fraction. 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

### 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 verbatim to participate in the expression evaluation, 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.

## Operations

Expr provides basic arithmetic operations, comparisons and logical operations.

The operations follow the general precedence rules for arithmetic, so  $A + B * C$  is calculated correctly. Comparison operations are done after the arithmetic operations and logical operations are done after comparisons. For detailed specification, see [Expr Language Reference](#).

Operations	Comments
+ - * /	Basic operators. When used, the value is converted to a number.
= !=	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 strings, then string comparison is used.  String 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.
( )	Parentheses can be used to group the results of operations prior to passing them to other operations.

## Advanced Constructs

### 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.

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.

Aggregate functions contain exactly one expression that is being aggregated, written in curly braces (`{ }`) after the function name.

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](#) for a complete list of available aggregate functions and modifiers.



Note that there is a `SUM( )` function and a `SUM{ }` aggregate function. You can always tell aggregate functions from the usual functions by the use of curly braces: `SUM{x}`.

## 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.

## 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 multiple lines of comment, start the comment with `/*` and end the comment with `*/`
- To add a single line of comment, begin the comment with `//`

Example:

```
/* This formula calculates the verbal assessment of issue's
progress.
   And this explanation is a comment that spans multiple lines. */
WITH total_time = time_spent + remaining_estimate :

// Progress is calculated based on time tracking. (This is a one-
line comment.)
WITH progress = IF(total_time > 0; time_spent / total_time) :

IF(progress > 0.5; "Great Progress!"; progress > 0.2; "Good
Progress"; "Needs Progress")
```

## Expr Reference Guides

### Expr Function Reference

All standard Expr functions are listed on this page, grouped by category.

### Notes About Functions

A function may take zero, one or more arguments. Some functions can take an unlimited number of arguments.

When a function expects a text or a numeric value as an argument and the actual type of value is different, the function will try to convert the value to the required type. If the conversion is not possible and the value is not empty (for example, it's impossible to convert "ABC" to a number), the result will be an error.

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 `undefined` value at its primary argument, the return value will also typically be `undefined`.

## Conditional Functions

### CASE

`CASE(Value; Match1; Result1; Match2; Result2; ...; DefaultOpt)`

Checks if the `Value` matches against several checks and returns a corresponding result.

- `Value` – value to check.
- `Match1, Match2, ..., MatchN` – 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](#) for details.
- `Result1, Result2, ..., ResultN` – values to return from the function, each value corresponds to the preceding `Match` parameter.
- `DefaultOpt` – optional default value, to be returned if none of the patterns match. If not specified, `undefined` is returned.

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 `DefaultOpt` 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.

- `Index` – numeric index, with 1 corresponding to `Value1`, 2 corresponding to `Value2` and so on.
- `Value1, Value2, ..., ValueN` – the values to pick from.

Examples:

- `CHOOSE(1; "A"; "B"; "C")` "A"
- `CHOOSE(2; "A"; "B"; "C")` "B"

## DEFINED

`DEFINED(Value)`

Checks if the value is defined. 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.

Examples:

- `DEFAULT(100; 500)` 100
- `DEFAULT(undefined; 500)` 500

## IF

`IF(Condition1; Result1; Condition2; Result2; ...; DefaultOpt)`

Checks one or several conditions and returns the result associated with the first true condition.

- `Condition1, Condition2, ..., Condition3` – the conditions 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` – results to be returned, each result corresponding to the preceding check.
- `DefaultOpt` – optional default value to be returned if none of the conditions are true. If omitted, undefined is returned.

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.

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; ErrorCodeOpt)`

Checks if calculating value produced an error. Returns true (1) if there was an error. If `ErrorCodeOpt` is specified, returns true only if the error was of the specified error code.

- `Value` – value to check.
- `ErrorCodeOpt` – optional error code. See [Expr Error Codes](#) for a list.

Examples:

- `ISERR("Ham") 0`
- `ISERR(1 / 0) 1`
- `ISERR(1 / 0, 4) 1 //Note: Error code 4 is an Arithmetic Error`

## Numeric Functions

### ABS

`ABS(Value)`

Calculates the absolute value of a number.

Examples:

- `ABS(5) 5`
- `ABS(-4) 4`

### CEILING

`CEILING(Value; N)`

Rounds value up to the  $N^{\text{th}}$  decimal place.

- `Value` – a number to round.

- $N$  – how many decimal places to round up to. Negative numbers round up to tens, hundreds, etc. Default value: 0 (round to an integer).

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  $N^{\text{th}}$  decimal place.

- `Value` – a number to round.
- $N$  – how many decimal places to round down to. Negative numbers round down to tens, hundreds, etc. Default value: 0 (round to an integer).

Examples:

- `FLOOR(1.678) 1`
- `FLOOR(12.34; 1) 12.3`
- `FLOOR(17.34; -1) 10`
- `FLOOR(-3.14) -4`

## MAX

`MAX(Value; ...)`

Selects the numerically largest value from all values passed as arguments. Undefined values are skipped. Text values that cannot be converted to a number will also be skipped.

Examples:

- `MAX(Due_Date; Updated_Date)`
- `MAX(0; -10; undefined; 10) 10`

## MIN

`MIN(Value; ...)`

Selects the numerically smallest value from all values passed as arguments. Undefined values are skipped. Text values that cannot be converted to a number will also be skipped.

Example:

- `MIN(0; -10; undefined; 10)` -10

## MOD

`MOD(A; N)`

Returns the remainder from dividing A by N.

- A – the dividend, must be an integer.
- N – the divisor, must be an integer.

Example:

- `MOD(17; 5)` 2

## POW

`POW(B; E)`

Produces B to the power of E ( $B^E$ ). Both values can be fractional.

- B – base
- E – exponent

Example:

- `POW(3; 3)` 27
- `POW(27; 1/3)` 3

## ROUND

`ROUND(Value; N)`

Produces a rounded value up to the N<sup>th</sup> decimal place.

- Value – a number to round.
- N – how many decimal places to round to. Negative numbers round to the nearest tens, hundreds, etc. Default value: 0 (round to an integer).

Examples:

- `ROUND(1.678)` 2
- `ROUND(12.34; 1)` 12.3
- `ROUND(12.34; -1)` 10

## SIGN

`SIGN(Value)`

Returns the sign of the Value (1 for positive, -1 for negative).

Examples:

- `SIGN(123)` 1
- `SIGN(0)` 0
- `SIGN(-123)` -1

## SQR

`SQR(Value)`

Returns the passed numerical value, squared.

Example:

- `SQR(5)` 25

## SQRT

`SQRT(Value)`

Returns the square root of the passed numerical value.

Example:

- `SQRT(25)` 5

## 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 (joins) strings together. Accepts any number of arguments. Ignores undefined values.

Example:

- `CONCAT(Reporter; ' => '; Assignee)`

## EXACT

`EXACT(A; B)`

Checks if text value A is exactly the same as text value B.

This comparison is case sensitive, which is different from comparing A with B using 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 string value.

- `Value` – string to get characters from.
- `N` – the number of characters to get. If `Value` contains fewer characters, all of them are returned.

Example:

- `LEFT("abc"; 2)` "ab"

## LEN

`LEN(Value)`

Returns the number of characters in a string value. If the value is not a string, it is converted to a string first.

Example:

- `LEN("abc")` 3

## LOWER

`LOWER(Value)`

Converts the string to lowercase. The locale of the current user is applied.

Example:

- `LOWER("HAM")` "ham"

## MATCH

`MATCH(Value; Pattern)`

Checks if the Value matches the Pattern. Returns `true` (1) or `false` (0).

- `Value` – the value to check.
- `Pattern` – pattern to check against. Can be an exact value, a wildcard expression or a regular expression. See [Expr Pattern Matching](#) for details.

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.

- `Value` – the string value to get a substring from.
- `Index` – the starting index of the part to retrieve, 1-based (first character is at index 1).
- `Count` – the number of characters to retrieve.

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.

- `Value` – a string value to repeat.
- `N` – the number of repetitions.

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 string.

- `Value` – the value to manipulate.
- `Pattern` – pattern to find. Can be an exact value, a wildcard expression or a regular expression. See [Expr Pattern Matching](#) for details.
- `Replacement` – an optional string to use instead of the matched parts. If omitted, the matched parts are removed.

Examples:

- `REPLACE("I like cats"; "CAT"; "DOG")` "I like DOGs"
- `REPLACE("Can you read this?"; "/[aeuiο]/")` "Cn y rd ths?"

## REPLACE\_AT

`REPLACE_AT(Value; Index; Count; Replacement)`

Replaces a specific part of the `Value` with `Replacement` string and returns the value.

- `Value` – the string to manipulate.
- `Index` – the starting index of the part to replace, 1-based (first character is 1, second is 2, etc.)
- `Count` – the number of characters to replace. When `Count` is 0, the `Replacement` string gets inserted at the `Index` position.
- `Replacement` – optional string to use instead of the replaced part. If omitted, the part will be deleted.

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.

- `Value` – string to get characters from.
- `N` – the number of characters to get. 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. Returns the index of the matched part (1-based), or `undefined` if not found.

- `Pattern` – the string or pattern to look for. Can be an exact value, a wildcard expression or a regular expression. See [Expr Pattern Matching](#) for details.
- `Value` – the string to search in.
- `Index` – optional parameter that provides an index to start searching at.

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`

## 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.

- `Value` – the string to take the part from.
- `From` – starting index, inclusive, 0 means the first character, `LEN(Value)-1` means the last character.
- `To` – 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`.

Examples:

- `SUBSTRING("Batman"; 0; 3)` "Bat"
- `SUBSTRING("Batman"; 3)` "man"

## TRIM

`TRIM(Value)`

Removes leading and trailing whitespace from the text.

Example:

- `TRIM(" Batman ")` "Batman"

## UPPER

`UPPER(Value)`

Converts the string to uppercase. The locale of the current user is applied.

Example:

- `UPPER("ham")` "HAM"

## Date and Time Functions

Date/time functions operate with a numeric representation of time. A moment in time is represented as a number of milliseconds since midnight, January 1st 1970, GMT. Negative values are allowed, representing times prior to January 1st 1970.

To display a result of a date/time calculation in a readable way, you need to either configure the [Formula Columns](#) to use a date/time format, or use one of the conversion functions to turn the value into a human-readable text.

Many of the date / time functions depend on the current user's time zone.

## DATE

`DATE(Text; LocaleOpt; TimeZoneOpt)`

Converts a text representation of a date to a number. The resulting timestamp will correspond to midnight of the specified date at the specified timezone.

- `Text` – the text value to convert.
- `LocaleOpt` – optional locale identifier, such as "fr\_FR". If not specified, user's locale is used.
- `TimeZoneOpt` – optional time zone identifier, such as "America/New\_York".

The conversion uses the standard formats for representing dates:

- Format "yyyy-MM-dd", like "2017-04-15".
- Standard formats for the specified locale.
- Jira formats, as specified in the Jira's system settings.

If conversion is unsuccessful, returns an error.

Examples:

- `DATE("2016-01-01")`
- `DATE("31/Dec/16")`
- `DATE("12/31/2016", "en_US", "America/New_York")`

## DATE\_ADD

`DATE_ADD(DateTime, Number, Unit)`

Adds the specified integer number of seconds, minutes, hours, days, months or years to the date or date/time value.

- `DateTime` – date or date/time value.
- `Number` – the integer number of units of time to add.
- `Unit` – a text value specifying the unit of time: "seconds", "minutes", "hours", "days", "months", "years"

Examples:

- `DATE_ADD(DATE("2016-01-31"), 1, "day")` `DATE("2016-02-01")`
- `DATE_ADD(DATE("2016-01-31"), 1, "month")` `DATE("2016-02-29")`
- `DATE_ADD(DATE("2016-02-29"), 1, "year")` `DATE("2017-02-28")`
- `DATE_ADD(DATETIME("2016-01-31 10:30:00"), 3, "hours")` `DATETIME("2016-01-31 13:30:00")`
- `DATE_ADD(DATETIME("2016-01-31 23:59:59"), 2, "minutes")` `DATETIME("2016-02-01 00:01:59")`

## DATE\_SET

`DATE_SET(DateTime, Number, Unit)`

Sets the specified part of the date or date/time to the specific integer value. Note that unlike `DATE_ADD` and `DATE_SUBTRACT`, you can specify additional units like "day\_of\_week".

- `DateTime` – date or date/time value.
- `Number` – the integer number to be set as the unit value in this date/time.

- **Unit** – a text value specifying the unit of time: "second", "minute", "hour", "day", "month", "year", "day\_of\_week".

#### Examples:

- `DATE_SET( DATE("2016-01-31"), 2017, "year" )` `DATE("2017-01-31")`
- `DATE_SET( DATE("2016-01-31"), 2, "month" )` `DATE("2016-02-29")`
- `DATE_SET( DATETIME("2016-02-29 15:30"), 10, "day" )` `DATETIME("2016-02-10 15:30")`
- `DATE_SET( DATE("2017-04-01"), 7, "day_of_week" )` `DATE("2017-04-02")`
- `DATE_SET( DATETIME("2016-01-31 10:30:00"), 0, "hour" )` `DATETIME("2016-01-31 00:30:00")`

## DATE\_SUBTRACT

`DATE_SUBTRACT( DateTime, Number, Unit )`

Subtracts the specified integer number of seconds, minutes, hours, days, months or years from the date or date/time value.

- **DateTime** – date or date/time value.
- **Number** – the integer number of units of time to subtract.
- **Unit** – a text value specifying the unit of time: "seconds", "minutes", "hours", "days", "months", "years"

#### Examples:

- `DATE_SUBTRACT( DATE("2016-02-01"), 1, "day" )` `DATE("2016-01-31")`
- `DATE_SUBTRACT( DATE("2016-02-29"), 1, "month" )` `DATE("2016-01-29")`
- `DATE_SUBTRACT( DATE("2017-02-28"), 1, "year" )` `DATE("2016-02-28")`
- `DATE_SUBTRACT( DATETIME("2016-01-31 10:30:00"), 3, "hours" )`  
`DATETIME("2016-01-31 07:30:00")`
- `DATE_SUBTRACT( DATETIME("2016-02-01 00:01:59"), 2, "minutes" )`  
`DATETIME("2016-01-31 23:59:59")`

## DAY

`DAY( DateTime )`

Returns the day of the month for the given date or date/time value. The result is calculated using the current user's time zone.

Example:

- `DAY( DATE( "2017-04-15" ) )` 15

## DAYS\_BETWEEN

`DAYS_BETWEEN( DateTime1, DateTime2 )`

Calculates the number of full days (24 hour periods) between two date or date/time values. Returns a negative value if `DateTime2` occurs earlier than `DateTime1`.

Examples:

- `DAYS_BETWEEN( DATE( "2017-01-01" ), DATE( "2017-02-01" ) )` 31
- `DAYS_BETWEEN( DATE( "2017-01-01" ), DATE( "2017-01-01" ) )` 0
- `DAYS_BETWEEN( DATE( "2017-01-01" ), DATE( "2016-01-01" ) )` -366
- `DAYS_BETWEEN( DATETIME( "2017-01-01 00:00" ), DATETIME( "2017-01-01 23:59" ) )` 0
- `DAYS_BETWEEN( DATETIME( "2017-01-01 23:59" ), DATETIME( "2017-01-02 23:58" ) )` 0
- `DAYS_BETWEEN( DATETIME( "2017-01-01 23:59" ), DATETIME( "2017-01-02 23:59" ) )` 1

## DATETIME

`DATETIME( Text; LocaleOpt; TimeZoneOpt )`

Converts a text representation of a date and time to a number. The resulting timestamp will correspond to the specified date and time at the specified timezone. If seconds are omitted, they will be set to zero.

- `Text` – the text value to convert.
- `LocaleOpt` – optional locale identifier, such as "fr\_FR". If not specified, user's locale is used.
- `TimeZoneOpt` – optional time zone identifier, such as "America/New\_York".

The conversion uses the standard formats for representing dates:

- Format "yyyy-MM-dd HH:mm:ss" and the same without seconds, like "2017-04-15 15:00" or "2017-12-31 23:59:59" (using 24-hour clock).
- Standard formats for the specified locale.
- JIRA formats, as specified in the JIRA's system settings.

If conversion is unsuccessful, returns an error.

**Examples:**

- `DATETIME("2016-01-01 00:01")`
- `DATETIME("31/Dec/16 3:15 pm")`
- `DATETIME("12/31/2016 3:15 PM", "en_US", "America/New_York")`

**END\_OF\_MONTH**

`END_OF_MONTH(DateTime)`

Sets the day in the date/time value to the end of the month. Does not change the time value.

**Example:**

- `END_OF_MONTH(DATE("2017-04-15")) DATE("2017-04-30")`

**FORMAT\_DATETIME**

`FORMAT_DATETIME(DateTime, Format, LocaleOpt, TimeZoneOpt)`

Advanced function to convert a date/time value into a text. Accepts an arbitrary format string and, optionally, locale and time zone settings. Does not depend on the current user's locale nor time zone.

- `DateTime` – the value to convert.
- `Format` – the format string. For all the options, please see [Java documentation for SimpleDateFormat](#).
- `LocaleOpt` – the optional locale identifier. If omitted or undefined, will use JIRA's system locale. (Not the user's locale!)
- `TimeZoneOpt` – the optional time zone identifier. If omitted or undefined, will use JIRA's system time zone. (Not the user's time zone!)

**Examples:**

- `FORMAT_DATETIME(DATE("2017-04-15"), "EEE, MMM d, `yy", "fr_FR")`  
`"sam., avr. 15, `17"`
- `FORMAT_DATETIME(DATETIME("2016-12-31 23:59"), "yyyy-MM-dd'T'HH:mm:ss")` `"2016-12-31T23:59:00"`

**HOUR**

`HOUR(DateTime)`

Returns the hour in the specified date/time value (from 0 to 23).

**Example:**

- `HOUR(DATETIME("2017-01-01 20:15"))` 20

## HOURS\_BETWEEN

`HOURS_BETWEEN(DateTime1, DateTime2)`

Calculates the number of full hours between two date/time values. Returns a negative value if `DateTime2` occurs earlier than `DateTime1`.

Examples:

- `HOURS_BETWEEN(DATE("2017-01-01"), DATE("2017-01-02"))` 24
- `HOURS_BETWEEN(DATETIME("2017-01-01 15:00"), DATETIME("2017-01-01 16:30"))` 1
- `HOURS_BETWEEN(DATETIME("2017-01-01 23:59"), DATETIME("2017-01-02 00:58"))` 0
- `HOURS_BETWEEN(DATETIME("2017-01-01 23:59"), DATETIME("2017-01-02 00:59"))` 1

## MAKE\_DATE

`MAKE_DATE(Year, Month, Day)`

Creates a date value based on the numbers defining year, month and day. The time is set to midnight in the user's time zone.

Example:

- `MAKE_DATE(2017, 12, 31)`

## MAKE\_DATETIME

`MAKE_DATETIME(Year, Month, Day, Hour, Minute, Second)`

Creates a date/time value based on the numbers defining year, month, day, hour, minute and second. The current user's time zone is used. The valid values for `Hour` are 0–23.

Example:

- `MAKE_DATETIME(2017, 12, 31, 23, 59, 59)`

## MINUTE

`MINUTE(DateTime)`

Returns the minutes in the specified date/time value (from 0 to 59).

Example:

- `MINUTE(DATETIME("2017-01-01 20:15"))` 15

## MONTH

`MONTH(DateTime)`

Returns the month in the specified date/time value (from 1 to 12).

Example:

- `MONTH(DATE("2017-04-15"))` 4

## MONTHS\_BETWEEN

`MONTHS_BETWEEN(DateTime1, DateTime2)`

Calculates the number of months between two date or date/time values. Returns a negative value if `DateTime2` occurs earlier than `DateTime1`.

Examples:

- `MONTHS_BETWEEN(DATE("2017-01-01"), DATE("2018-01-01"))` 12
- `MONTHS_BETWEEN(DATE("2017-01-31"), DATE("2017-02-28"))` 0
- `MONTHS_BETWEEN(DATE("2017-02-28"), DATE("2017-04-28"))` 2
- `MONTHS_BETWEEN(DATE("2017-01-01"), DATE("2016-12-01"))` -1

## NOW

`NOW()`

Returns the current date and time.

Example:

- `NOW()`

## PARSE\_DATETIME

`PARSE_DATETIME(Text, Format, LocaleOpt, TimeZoneOpt)`

Advanced function to convert a text into a date or date/time value. Accepts an arbitrary format string and, optionally, locale and time zone settings. Does not depend on the current user's locale nor time zone.

- `Text` – the value to convert.
- `Format` – the format string. For all the options, please see [Java documentation for SimpleDateFormat](#).

- `LocaleOpt` – the optional locale identifier. If omitted or undefined, will use JIRA's system locale. (Not the user's locale!)
- `TimeZoneOpt` – the optional time zone identifier. If omitted or undefined, will use JIRA's system time zone. (Not the user's time zone!)

Examples:

- `PARSE_DATETIME("sam., avr. 15, `17", "EEE, MMM d, `yy", "fr_FR")`  
`DATE("2017-04-15")`
- `PARSE_DATETIME("2016-12-31T23:59:00", "yyyy-MM-dd'T'HH:mm:ss")`  
`DATETIME("2016-12-31 23:59")`

## SECOND

`SECOND(DateTime)`

Returns the seconds in the specified date/time value.

Example:

- `SECOND(DATETIME("2017-04-15 15:30:59"))` 59

## START\_OF\_MONTH

`START_OF_MONTH(DateTime)`

Sets the day in the date/time value to the first day of the month.

Example:

- `START_OF_MONTH(DATE("2017-04-15"))` `DATE("2017-04-01")`

## TODAY

`TODAY()`

Returns the current date with time set to midnight according to the current user's time zone.

Example:

- `TODAY()`

## TRUNCATE\_TIME

`TRUNCATE_TIME(DateTime)`

Removes the time value from the date/time, setting it to midnight in the current user's time zone.

Example:

- `TRUNCATE_TIME(DATETIME("2017-01-01 15:15")) DATE("2017-01-01")`

## TRUNCATE\_TO\_HOURS

`TRUNCATE_TO_HOURS(DateTime)`

Removes the minutes, seconds and milliseconds from the date/time, setting it to the last even hour in the current user's time zone.

Example:

- `TRUNCATE_TO_HOURS(DATETIME("2017-01-01 15:15")) DATE("2017-01-01 15:00")`

## TRUNCATE\_TO\_MINUTES

`TRUNCATE_TO_MINUTES(DateTime)`

Removes the seconds and milliseconds from the date/time, setting it to the last even minute.

Example:

- `TRUNCATE_TO_MINUTES(DATETIME("2017-01-01 15:15:15")) DATE("2017-01-01 15:15:00")`

## TRUNCATE\_TO\_SECONDS

`TRUNCATE_TO_SECONDS(DateTime)`

Removes the milliseconds from the date/time.

Example:

- `TRUNCATE_TO_SECONDS(NOW())`

## WEEKDAY

`WEEKDAY(DateTime)`

Returns the number of the day in the week, following ISO-8601 standard (1 – Monday, 7 – Sunday).

Example:

- `WEEKDAY(DATE("2017-04-23")) 7`

## YEAR

`YEAR(DateTime)`

Returns the year in a date or date/time value as a number.

Example:

- `YEAR( DATE( "2017-04-23" ) )` 2017

## YEARS\_BETWEEN

`YEARS_BETWEEN( DateTime1, DateTime2 )`

Calculates the number of years between two date or date/time values. Returns a negative value if `DateTime2` occurs earlier than `DateTime1`.

Examples:

- `YEARS_BETWEEN( DATE( "2017-01-01" ), DATE( "2018-01-01" ) )` 1
- `YEARS_BETWEEN( DATE( "1703-05-27" ), DATE( "2017-04-23" ) )` 313
- `YEARS_BETWEEN( DATE( "2017-06-01" ), DATE( "2018-05-31" ) )` 0

## 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. 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. 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. 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. May return a fractional number of seconds.

Example:

- `CALENDAR_SECONDS(DURATION("1h"))` 3600

## DURATION

`DURATION(Text)`

Converts a text representation of a calendar duration to a number. The format is provided by Jira – the text may be several numbers, each number followed by a symbol to specify the time unit: w for weeks, d for days, h for hours and m for minutes.

Note that this function ignores Jira's settings for work time, so `DURATION("1w") = DURATION("7d")` and `DURATION("1d") = DURATION("24h")`.

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.

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. (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 format is provided by Jira – the text may be several numbers, each number followed by a symbol to specify the time unit: `w` for weeks, `d` for days, `h` for hours and `m` for minutes.

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")`.

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. (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

## Miscellaneous Functions

### ME

ME()

Returns the user key of the current user.

Example:

- `IF(ME() = "admin"; "You're admin!")`

### NUMBER

NUMBER(Value)

Converts value to number. This function is rarely needed, because conversion to number happens automatically when needed.

Example:

- `NUMBER("1.234")` 1.234

### TEXT

TEXT(Value)

Converts value to text. This function is rarely needed, because conversion to text happens automatically when needed.

Example:

- `TEXT(1.234)` "1.234"

## Aggregate Function Reference

All standard aggregate functions and available modifiers are listed on this page.

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.



In Structure versions prior to 6.0, it was not possible to include both upward-looking and downward-looking aggregate functions within the same formula. When using one of the two upward-looking aggregate functions, PARENT and JOIN (when used with an upward-looking modifier), you could not include any of the other aggregate functions listed above.

Starting with Structure 6, this limitation is removed.

## Aggregation Functions

### 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)
▼ <input checked="" type="checkbox"/> T1	3	2
<input checked="" type="checkbox"/> T1.1	2	2
▼ <input checked="" type="checkbox"/> T1.2		1
<input checked="" type="checkbox"/> 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, #children, #leaves, #strict, #preceding, #levels (together with #preceding), #baseLevel (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)
▼ <input checked="" type="checkbox"/> T1	3	3
<input checked="" type="checkbox"/> T1.1	2	1
▼ <input checked="" type="checkbox"/> T1.2		1
<input checked="" type="checkbox"/> 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, #children, #leaves, #strict, #truthy, #preceding, #levels (together with #preceding), #baseLevel (together with #preceding).

## 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` modifier is set, join appends the values for children, wrapping them into characters (braces by default).
- Wrapping characters can be set by `#beforeChildren` and `#afterChildren` (see example for `#subtree` to see how it works).

Summary	x	JOIN(x)
▼ <input checked="" type="checkbox"/> T1	3	3
<input checked="" type="checkbox"/> T1.1	2	3,2
▼ <input checked="" type="checkbox"/> T1.2		3,?
<input checked="" type="checkbox"/> T1.2.1	1	3,?,1

Accepts modifiers: `#ancestors`, `#subtree`, `#children`, `#leaves`, `#strict`, `#reverse`, `#separator`, `#beforeChildren`, `#afterChildren`, `#fromLevel`, `#toLevel`, `#distinct`.

## 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)
▼ <input checked="" type="checkbox"/> T1	3	3
<input checked="" type="checkbox"/> T1.1	2	2
▼ <input checked="" type="checkbox"/> T1.2		1
<input checked="" type="checkbox"/> T1.2.1	1	1

Accepts modifiers: `#children`, `#leaves`, `#strict`.

## 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)
▼ <input checked="" type="checkbox"/> T1	3	1
<input checked="" type="checkbox"/> T1.1	2	2
▼ <input checked="" type="checkbox"/> T1.2		1
<input checked="" type="checkbox"/> T1.2.1	1	1

Accepts modifiers: `#children`, `#leaves`, `#strict`.

## PARENT

Parent extracts the value from the parent row or from an ancestor row by a specified level.

Summary	X	PARENT(X)
▼ <input checked="" type="checkbox"/> T1	3	
<input checked="" type="checkbox"/> T1.1	2	3
▼ <input checked="" type="checkbox"/> T1.2	3	
<input checked="" type="checkbox"/> T1.2.1	1	

Accepts modifier: [#level](#).

## SUM

Sum calculates a numerical total for the values calculated for the item and/or its sub-items.

Summary	X	SUM(X)
▼ <input checked="" type="checkbox"/> T1	3	6
<input checked="" type="checkbox"/> T1.1	2	2
▼ <input checked="" type="checkbox"/> T1.2	1	1
<input checked="" type="checkbox"/> T1.2.1	1	1

Other variations of `SUM` allow different types of aggregation:

- `SUM{ }` 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

Note that when the value of the expression under aggregation is not numeric (and cannot be [converted](#) 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](#), [#children](#), [#leaves](#), [#strict](#).

## 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
▼ <input checked="" type="checkbox"/> T1	3	3<[2, ?<{1}>>
<input checked="" type="checkbox"/> T1.1	2	2
▼ <input checked="" type="checkbox"/> T1.2		?<{1}>
<input checked="" type="checkbox"/> T1.2.1	1	1

```
JOIN#subtree#beforeChildren=" <
{ "#afterChildren=" } > " {X}
```

Can be used with: [join](#).

### #all

Tells the aggregate function to include duplicate items. By defaults, aggregate functions ignore duplicate items.

### Example

Summary	X	SUM(X)	SUM#all(X)	COUNT(X)	COUNT#all(X)
▼ <input checked="" type="checkbox"/> T1	2	6	6	1	3
<input checked="" type="checkbox"/> T1.1	2	2	2	1	1
<input checked="" type="checkbox"/> T1.1	2	2	2	1	1
<input checked="" type="checkbox"/> T1.1	2	2	2	1	1

```
SUM#all{X}
COUNT#all
{X}
```

Can be used with: [sum](#), [count](#), [avg](#).

### #ancestors

Only process ancestors of the current row. This is the default behavior for [join](#) and [parent](#).

Can be used with: [join](#).

### #baseLevel

Can be used with [SUM](#) 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](#), [count](#), [sum](#) (together with `#preceding`).

## #beforeChildren

See [#afterChildren](#).

## #children

Only process direct children of the current row.

### Example

Summary	X	JOIN#children(X)	SUM#children(X)
▼ <input checked="" type="checkbox"/> T1	3	2, ?	2
<input checked="" type="checkbox"/> T1.1	2		
▼ <input checked="" type="checkbox"/> T1.2		1	1
<input checked="" type="checkbox"/> T1.2.1	1		

```
JOIN#children{X}
SUM#children{X}
```

Can be used with: [sum](#), [count](#), [avg](#), [join](#), [min](#), [max](#).

## #depth

Same as [#level](#) modifier.

## #distinct

Makes [join](#) only concatenate distinct values. A duplicate value won't be added more than once if this modifier is on.

Modifiers [#beforeChildren](#) and [#afterChildren](#) don't work when this option is on.

### Example

---

Summary	JOIN#distinct{project}	JOIN#subtree#distinct{project}	JOIN{project}	JOIN#su
▼ <input checked="" type="checkbox"/> t1	PRJ	PRJ	PRJ	PRJ(PR
▼ <input checked="" type="checkbox"/> t2	PRJ	PRJ	PRJ, PRJ	PRJ(PR
▼ <input checked="" type="checkbox"/> t3	PRJ	PRJ	PRJ, PRJ, PRJ	PRJ(PR
<input checked="" type="checkbox"/> t4	PRJ	PRJ	PRJ, PRJ, PRJ, PRJ	PRJ

Can be used with: [join](#).

### #fromDepth

Same as [#fromLevel](#).

### #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](#).

### Example

Summary	X	JOIN#fromDepth=-1(X)	JOIN#fromDepth=2(X)
▼ <input checked="" type="checkbox"/> T1	1	1	
▼ <input checked="" type="checkbox"/> T2	2	1, 2	2
▼ <input checked="" type="checkbox"/> T3		2, ?	2, ?
<input checked="" type="checkbox"/> T4	4	?, 4	2, ?, 4

```
JOIN#fromLevel=-1{X}
JOIN#fromLevel=2{X}
```

Can be used with: [join](#).

### #leaves

Only process leaves (items without children) in the subtree of the current row.

### Example

Summary	X	JOIN#leaves(X)	SUM#leaves(X)
▼ <input checked="" type="checkbox"/> T1	3	2, 1	3
<input checked="" type="checkbox"/> T1.1	2	2	2
▼ <input checked="" type="checkbox"/> T1.2		1	1
<input checked="" type="checkbox"/> T1.2.1	1	1	1

```
JOIN#leaves{X}
SUM#leaves{X}
```

Can be used with: [sum](#), [count](#), [avg](#), [join](#), [min](#), [max](#).

## #level

When used with [PARENT](#), 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](#) with the `#preceding` modifier, this specifies the level at which the values should be aggregated.

### Example

Summary	X	-1	-2	1	2
▼ <input checked="" type="checkbox"/> T1	3			3	
<input checked="" type="checkbox"/> T1.1	2	3		3	2
▼ <input checked="" type="checkbox"/> T1.2		3		3	
<input checked="" type="checkbox"/> 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](#), [sum](#) (together with `#preceding`), [count](#) (together with `#preceding`), [avg](#) (together with `#preceding`).

## #levels

Can be used with [SUM](#) 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](#) modifier, only values at levels that are deeper than the base level will be counted.

You can also use [#level](#) instead of #levels.

Can be used with: [avg](#), [count](#), [sum](#) (together with [#preceding](#)).

### #reverse

Reverses the order of row processing.

#### Example

Summary	x	JOIN#reverse(X)
▼ <input checked="" type="checkbox"/> T1	3	3
<input checked="" type="checkbox"/> T1.1	2	2,3
▼ <input checked="" type="checkbox"/> T1.2		?,3
<input checked="" type="checkbox"/> T1.2.1	1	1,?,3

```
JOIN#
rever
se{X}
```

Can be used with: [join](#).

### #separator

Defines the separator for string joining. This modifier has a string parameter. The default is ", ".

#### Example

Summary	x	JOIN#separator=">"(X)
▼ <input checked="" type="checkbox"/> T1	3	3
<input checked="" type="checkbox"/> T1.1	2	3->2
▼ <input checked="" type="checkbox"/> T1.2		3->?
<input checked="" type="checkbox"/> T1.2.1	1	3->?->1

```
JOIN#
separ
ator=
">" {
X}
```

Can be used with: [join](#).

### #strict

Do not process the current row item as part of the aggregation.

Cannot be used together with [#children](#), [#ancestors](#) or [#leaves](#), since these already exclude the current row.

#### Example

Summary	x	JOIN#strict(X)	SUM#strict(X)
▼ <input checked="" type="checkbox"/> T1	3	2, ?(1)	3
<input checked="" type="checkbox"/> T1.1	2		
▼ <input checked="" type="checkbox"/> T1.2		1	1
<input checked="" type="checkbox"/> T1.2.1	1		

```
JOIN#
strict{X}
SUM#strict{X}
```

Can be used with: [sum](#), [count](#), [avg](#), [join](#), [min](#), [max](#).

### #subtree

Process the whole subtree of the current row. This is the default behavior for [sum](#), [count](#), [avg](#), [min](#), [max](#).

#### Example

Summary	x	JOIN#subtree(X)	SUM(X)
▼ <input checked="" type="checkbox"/> T1	3	3(2, ?(1))	6
<input checked="" type="checkbox"/> T1.1	2	2	2
▼ <input checked="" type="checkbox"/> T1.2		?(1)	1
<input checked="" type="checkbox"/> T1.2.1	1	1	1

```
JOIN#
subtree{X}
```

Can be used with: [join](#).

### #toDepth

Same as [#toLevel](#).

### #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](#).

### Example

Summary	X	JOIN#toDepth=-1(X)	JOIN#toDepth=2(X)
▼ <input checked="" type="checkbox"/> T1	1		
▼ <input checked="" type="checkbox"/> T2	2	1	1, 2
▼ <input checked="" type="checkbox"/> T3		1, 2	1, 2
<input checked="" type="checkbox"/> T4	4	1, 2, ?	1, 2

```
JOIN#
toLev
el=-1
{X}
JOIN#
toLev
el=2
{X}
```

Can be used with: [join](#).

### #truthy

Only count row if the subexpression produces a [truthy value](#) .

### Example

Summary	X	COUNT#truthy(X)
▼ <input checked="" type="checkbox"/> T1	0	2
<input checked="" type="checkbox"/> T1.1	2	1
▼ <input checked="" type="checkbox"/> T1.2		1
<input checked="" type="checkbox"/> T1.2.1	1	1

```
COUNT
#trut
hy{X}
```

Can be used with: [count](#).

## Expr Language Reference

Expr language defines expressions, which are evaluated in the context of an item within a structure. This article describes the syntax of the language and the rules that govern the evaluation.

### Conventions

- Similarity to Excel formula language was a design goal, so if you are unsure how Expr behaves, think Excel.
- The language is case-insensitive.

- Whitespace 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.
- Currently, language constructs support only English letters and a few punctuation symbols. However, values can contain any Unicode symbols.

## 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.

## Values

All expressions, when evaluated, produce either a value or an error. All values in Expr are either numbers, text or a special value called *undefined*.

## Undefined

Undefined value is represented by the word `undefined`.

Undefined value is used when the variable value is not specified. For example, variable `Assignee` has value `undefined` if the issue is unassigned.

Functions can return this value 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.

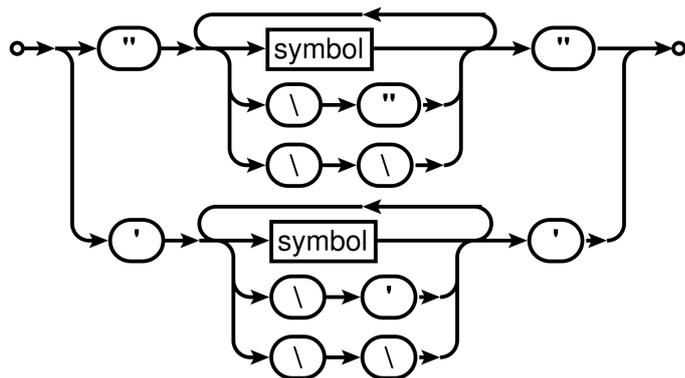
If the text value itself contains quotes, you'll need to insert a backslash (`\`) before them.

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\*.



## Numbers

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 Columns](#) 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 quantities:  $X$ , 0, and 618.

Group separators are not supported, so 100 000 is not a literal representation of number 100000.

**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.

## 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](#) 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](#); 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

## Falsy and Truthy Values

A value is *falsy* if it is:

- undefined,
- number 0,
- empty text value (" " or ' '), or a text value that contains only space characters.

All other values are *truthy*. By convention, when predefined functions or logical operators need to construct a truthy value, they use number 1.

## Variables and functions

Other kinds of expressions are variables and function calls.

### Identifiers

An identifier consists of letters (Latin alphabet only: a-z, A-Z), digits (0-9), dot (.) or underscore (\_) characters. The first character must be a letter or an underscore.

### Variables

Variables are represented by identifiers. Variables are defined in the [Formula Columns](#) settings and are mapped to a Jira field or other attribute of an item.

Conceptually, you can think of a variable as the cell of some column for the item, in the context of which the expression is evaluated. As such, it might or might not have a value, and that value can be either textual or numeric. Each variable is resolved to a value once during the expression evaluation. If the variable cannot be resolved, its value is `undefined`.

### Local Variables

Local variables are similar to Variables, but they are not mapped to the item's attribute or Jira field, but rather defined and calculated right in the expression.

The declaration syntax is the following:

```
WITH <local_variable_name> = <expression> :  
<expression_with_local_variable>
```

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:

- `<expression_with_local_variable>` may start with another local variable definition, so you can introduce many local variables. 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.

## Function Calls

A function consumes zero or more values, and can produce a 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.

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.

## Aggregate Function Calls

An aggregate function call takes an expression and calculates it for all sub-items (or for another sub-set of the structure, as defined in the function's documentation).

An aggregate function may have one or more modifiers that govern aspects of function execution. Each modifier starts with hash sign ("#"), then comes the name (an identifier), an optional equal sign ("=") and a value, which can be a string or numeric constant. If a value is omitted, it is assumed to be 1 (a representation of *true* in Expr).

An aggregate function must be followed by the expression in curly braces ("{}"), which provides the values being aggregated.

You can use whitespace between any elements of the aggregate function calls.

Examples:

- `SUM{x}`

- `SUM#all{x}`
- `SUM#all#leaves{x}`
- `JOIN#separator=", "{key}`
- `JOIN #separator=", " #fromDepth=0 #toDepth=-1 { Key }`

Not all modifiers will work with every aggregate function. Using an incompatible modifier will result in an error. To learn more about available modifiers and their restrictions, see [Aggregation Modifiers](#).

## Single-argument operators

Expressions with a single-argument (or *unary*) operator have the following syntax: `<op> <expression>`.

`<expression>` can be any Expr language expression in parentheses. If it is a literal value representation, a variable or a function call, parentheses are optional.

If `<expression>` evaluation produces an error, the operator also produces an error.

## NOT

Instead of `NOT`, an exclamation mark (!) can also be used.

The operator produces 0 if `<expression>` evaluates to a truthy value, and 1 otherwise.

## + -

The operator first attempts to convert the value of `<expression>` to a number. If conversion succeeds, + produces this number, and - produces the negated number. If conversion fails, and the value of `<expression>` is falsy, it produces `undefined`. Otherwise, it produces an error.

## Logical and arithmetic operators

Two or more expressions can be combined using operators: `<expression1> <operator> <expression2>`. If any subexpression produces an error, the operator produces the same error.

## Logical operators

OR (`||`, `|`)

AND (`&&`, `&`)

OR examines each expression from left to right and produces the value of the first expression that evaluates to a truthy value. If no expression evaluates to a truthy value, it returns `undefined`. Once a truthy expression is found, no other expressions are evaluated. This prevents unnecessary computations and protects against producing an error if any of the subsequent expressions produce an error.

AND works in much the same way, except that it is looking for the first *false* value.

Examples (assuming the default variable assignment):

- `assignee || "UNASSIGNED"` – This will produce either the issue's assignee user key or (if the issue is unassigned) the text value "UNDEFINED".
- `!assignee && status = "OPEN"` – This will produce 1 if the issue is unassigned and in status OPEN, and 0 otherwise.

## Comparison operators

All comparison operators:

- Produce 0 or 1
- Can work only on two arguments
- Start with evaluating both expressions
- Have the same precedence

Equality: `=` (`==`).

If both values are numbers, returns 1 if they are equal.

If both values are text, returns 1 if they are equal, ignoring differences in letter forms and leading and trailing whitespace (thus `" cote "` = `"côte"`).

If both values are undefined, returns 1.

In all other cases, returns 0.



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](#) 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

Inequality: `<>` (`! =`)

Works in the same way as the equality operator, but returns 0 when both values are equal or undefined, 1 when they are not.

Ordering

`<` (less than)

`>` (greater than)

`<=` (less than or equal)

`>=` (greater than or equal)

All operators work on numbers, producing the result of their comparison.

If either of the values is text, the operator attempts to convert it to number. If the conversion fails, the operator behaves as if the corresponding value was undefined.

If any value is undefined, strict operators (`<`, `>`) produce 0. Non-strict (`<=`, `>=`) produce 0, unless *both* values are undefined (because they are equal).

## Arithmetic operators

Arithmetic operators are: addition (`+`), subtraction (`-`), multiplication (`*`) and division (`/`).

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

## Precedence of operators

Precedence defines which operators evaluate first: if operator A has lesser precedence than B, then in expression `<expression1> A <expression2> B <expression3>` first B is evaluated, then A.

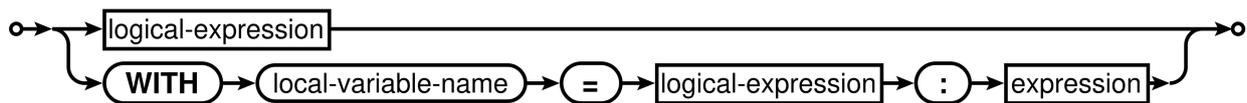
Single-argument operators are always evaluated first. Other operators in Expr language have the following precedence:

5 (highest)	* /
4	+ -
3	= <> < > <= >=
2	AND
1 (lowest)	OR

## Railroad diagrams

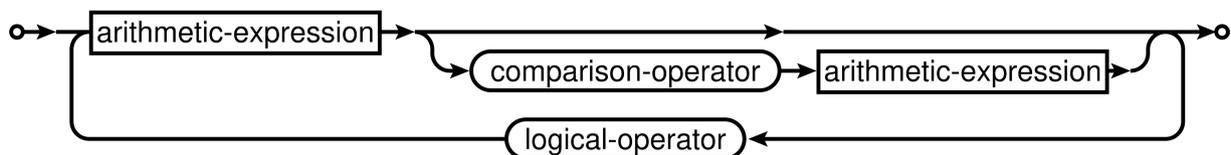
These diagrams display the complete syntax of Expr language.

### expression



local-variable-name is an [identifier](#).

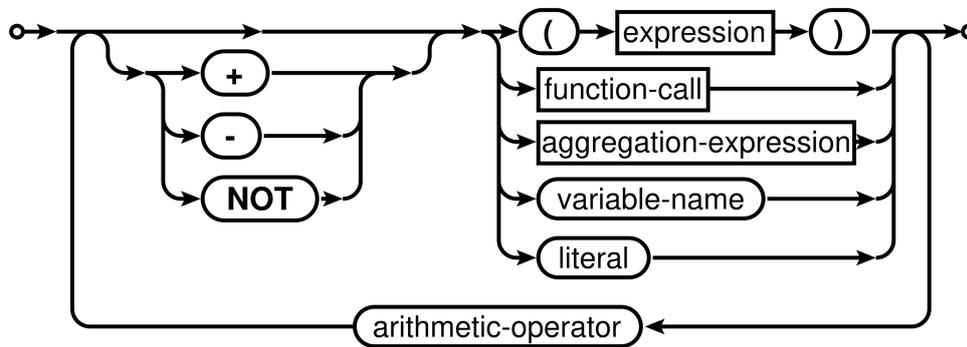
### logical-expression



comparison-operator is one of these: = <> < > <= >=.

logical-operator is one of these: AND OR.

## arithmetic-expression

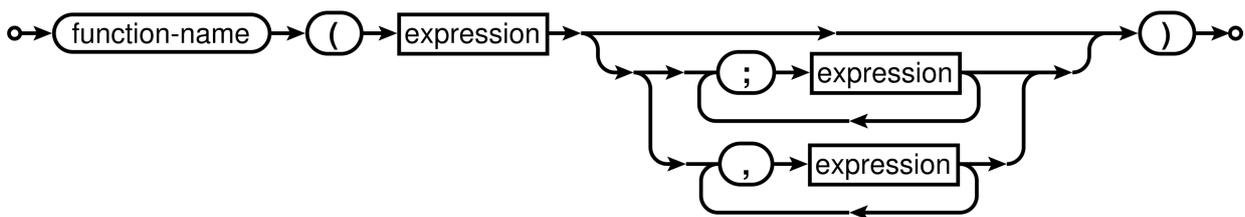


variable-name is an [identifier](#).

literal is either a [number literal](#), a [text](#) or [UNDEFINED](#).

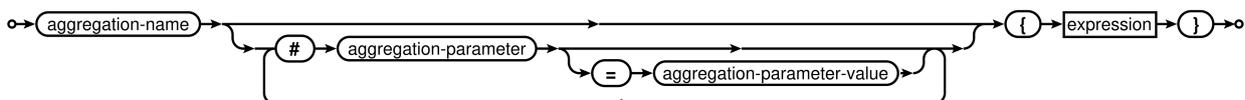
arithmetic-operator is one of these: + - \* /.

## function-call



function-name is an [identifier](#).

## aggregation-expression



aggregation-name and aggregation-parameter are [identifiers](#).

aggregation-parameter-value is either a [text](#) or a [number literal](#) with optional sign (either + or -).

## Expr Pattern Matching

Expr language allows you to check text values against a certain pattern, when using the [MATCH](#), [CASE](#), [REPLACE](#) or [SEARCH](#) function.

There are three types of patterns that can be used: [Exact Matching](#), [Wildcard Matching](#), and [Regular Expression Matching](#).

## 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 a full documentation about the regular expression language, see [Java documentation for Pattern](#).

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 "/" and ".". (These characters are removed, as they are not a part of the pattern.)

## 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](#) function.

Error Code	Name	Displayed As	Description
1	Parse Error	???	The expression is invalid. To fix, review and edit the expression.
2	Unknown Function	FUNC?	The expression contains a function that is not available or does not exist. To fix, review used functions in the expression to see if there are any typos.
3	Bad Number of Arguments	ARGS?	A function is used with an incorrect number of arguments. To fix, review the expression and see if all functions are called with a correct number of arguments.
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.

Error Code	Name	Displayed As	Description
			To fix, review your formula to ensure you're not dividing by zero. (To avoid division by zero, use the <a href="#">IF</a> function.) If this does not work, try separately calculating parts of the formula to identify the error.
5	Variable Error	<b>VAR!</b>	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.
6	Function Execution Error	<b>FUNC!</b>	A function suffered an internal error. Refer to logs for details.
7	Value Conversion Error	<b>VALUE?</b>	A value could not be converted to the desired format. Check the formula. Most likely a non-convertible text value was submitted to a function as a number.
8	Malformed Regex	<b>REGEX?</b>	A text with invalid regular expression was passed to a matching function. Check the regular expressions that you use. See <a href="#">Expr Pattern Matching</a> for details.
9	Internal Error	<b>ERROR!</b>	There's an internal problem with Jira or Structure. Refer to the logs for details.
10	Invalid Value	<b>VALUE!</b>	An invalid value was passed as a function argument. For example, using an unknown unit of time would produce this error. Check arguments for the functions that expect specific values.
11	Aggregation Error	<b>AGGR?</b>	The formula contains an unknown aggregate function or an invalid aggregate function modifier. Check that your <a href="#">aggregate function and modifiers</a> are valid and spelled correctly.

## 2.6 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](#)
- in the Search Area on the Structure Widget — see [Search](#)
- in the [S-JQL Filter generator](#)
- in the workflow validator or condition — see [Workflow Integration](#)

### 2.6.1 Get Started with S-JQL

We recommend starting with the [S-JQL Cookbook](#), which contains a number of common examples that can be used as is or modified to meet your specific needs.

### 2.6.2 Master S-JQL

For a comprehensive description of the language and `structure()` JQL function, consult the [S-JQL Reference](#).

### 2.6.3 S-JQL Cookbook

Here are the most common examples of using S-JQL.

#### 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](#) and run the following query:

```
issue in structure("My todo list")
```

If you want to find issues added to the [Default Structure](#), you can omit the structure name:

```
issue in structure()
```

[^ up to the list of examples](#)

## 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](#) 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](#)

## 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](#) 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](#) and type this query in the [Search Area](#) in the JQL mode.

Also, you can type only the last part of the query if you use [S-JQL search mode](#):

```
issueOrAncestor in [type = Epic and status = Assigned]
```

[^ up to the list of examples](#)

## 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](#) capability or open the structure on the [Structure Board](#) and use its [Search Area](#) 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](#) on the Structure Board:

```
[type = 'Test Case'] and ancestor in [type = Story and sprint in openSprints()]
```

[^ up to the list of examples](#)

## 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](#) 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](#) capability or open the structure on the [Structure Board](#) and use its [Search Area](#) 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](#) on the Structure Board:

```
child of [priority = Critical]
```

[^ up to the list of examples](#)

### 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](#)

### 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](#) with the following JQL:

```
issue in structure("Project work breakdown", "root and  
descendants in [resolution is empty]")
```

[^ up to the list of examples](#)

## 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](#) on the [Structure Board](#), run the following [JQL search](#):

```
issue in structure("Planning", "[type = Task] and parent not in  
[type in (Epic, Story)]")
```

[^ up to the list of examples](#)

## 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](#) on the [Structure Board](#), run the following [S-JQL search](#):

```
[resolution is not empty] and child in [resolution is empty]
```

[^ up to the list of examples](#)

## 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](#), 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](#). When you paste the query given below into the [Search Area](#) (ensure that the [JQL mode](#) 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](#)

### 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](#) on the [Structure Board](#) to run the specific queries (ensure that the [S-JQL mode](#) is selected.) Toggle the [Filter](#) 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](#)

## 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](#) 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](#)

### 2.6.4 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](#) capability of JIRA.

List of Structured JQL topics:

### 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:

```
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` will match the first instance but not the second one.

This difference is visible when you are filtering in the Structure Widget (see [Filter](#)). However, [structure\(\) JQL function](#) 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`.

## 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](#)

## 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](#) 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
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 Desk plugin is available)
```

[Structure.Pages](#) 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`.  
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 it as a fully qualified name. Only if nothing is found, it tries to interpret it as a shortened form.

**i** 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](#) in relational constraints, which are discussed further.

[^ up to the list of S-JQL topics](#)

## 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](#)

## 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 operator subConstraint`. 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](#)

## 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](#) 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:

<code>childOrSelf</code>	<code>childOrIssue</code>
<code>parentOrSelf</code>	<code>parentOrIssue</code>
<code>descendantOrSelf</code>	<code>descendantOrIssue</code>
<code>ancestorOrSelf</code>	<code>ancestorOrIssue</code>
<code>siblingOrSelf</code>	<code>siblingOrIssue</code>
<code>prevSiblingOrSelf</code>	<code>prevSiblingOrIssue</code>
<code>nextSiblingOrSelf</code>	<code>nextSiblingOrIssue</code>



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](#)

## 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
<code>in</code>	<i>add to result set</i>	<i>satisfies subConstraint</i>
<code>of</code>	<i>satisfies subConstraint</i>	<i>add to result set</i>



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](#)

## Sub-constraints

Any constraint can be used as a sub-constraint, whether basic, relational, or a [combination of those](#).

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.

<code>child is empty</code>	matches all items that have no children (equivalent of <code>leaf</code> )
<code>child is not empty</code>	matches all items that have at least one child (equivalent of <code>not leaf</code> )
<code>child of empty</code>	matches all items that are not children of other items (equivalent of <code>root</code> )

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](#)

### 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](#)

## 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]
```

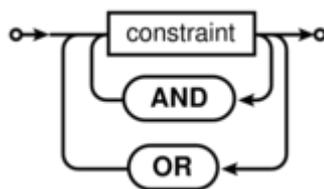
[^ up to the list of S-JQL topics](#)

## Railroad diagrams

As a final piece of reference, here's the S-JQL syntax in the form of [railroad diagrams](#).

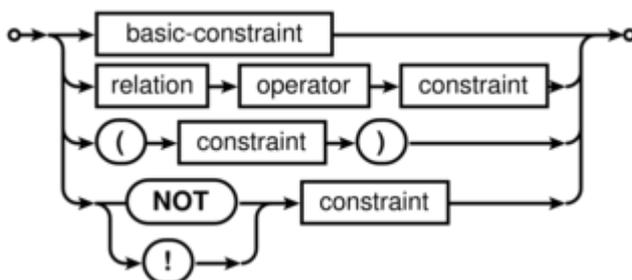
**i** S-JQL keywords are not case-sensitive, and all underscores in keywords are optional.

### structure-query

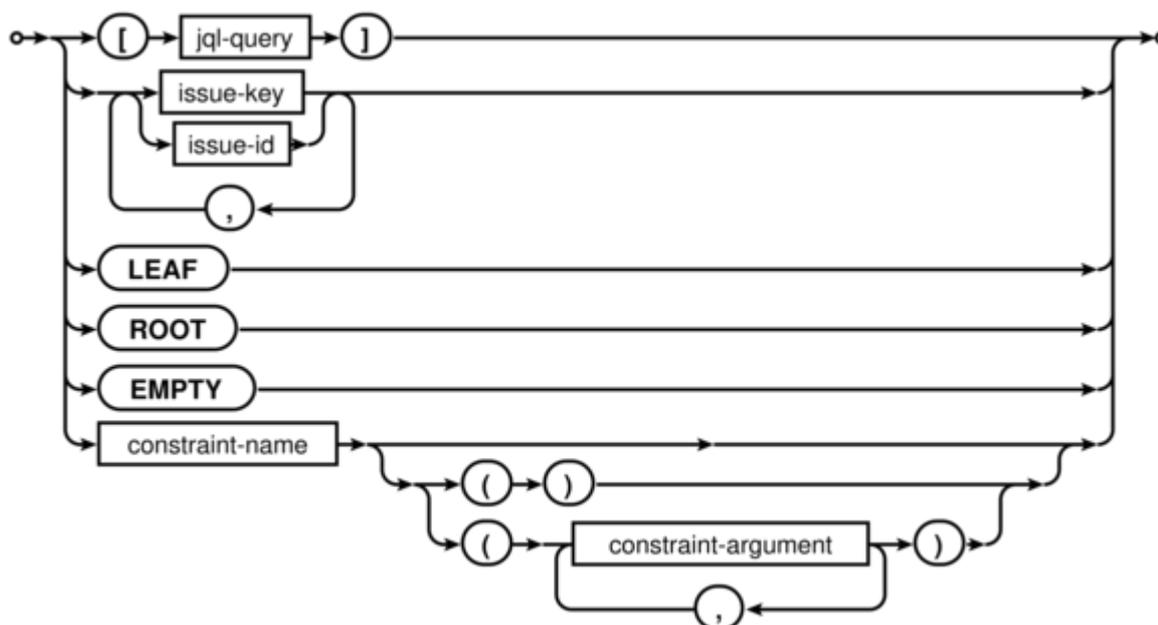


**i** S-JQL admits using `&&` and `&` in place of `AND`, as well as `||` and `|` in place of `OR`.

### constraint



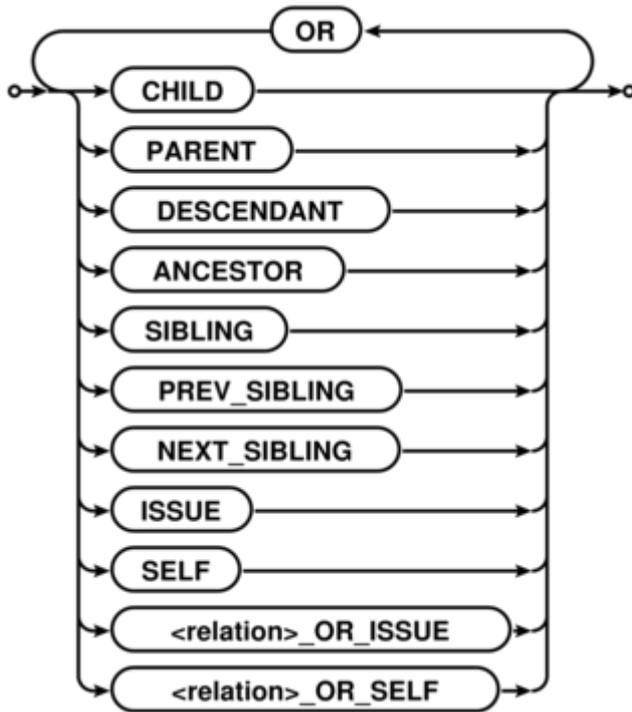
## basic-constraint



- `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: `\\`.

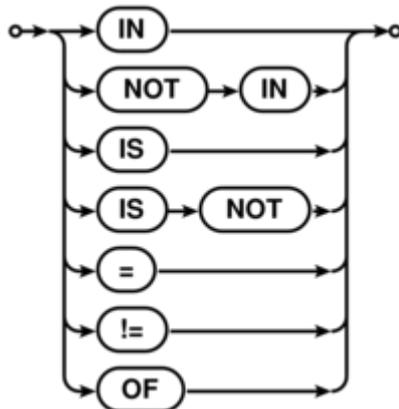
See also [Function constraint - Syntax](#).

relation



**i** S-JQL admits using `||` and `|` in place of OR.

### operator



[^ up to the list of S-JQL topics](#)

### 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
|
||
```

### 2.6.5 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](#) and [Advanced Searching Functions](#).

**i** If a user does not have [access to structure](#), 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(structureNameopt, structureQueryopt)
```

Function arguments:

<b>structureName</b>	<i>Optional</i>	The name of the structure. If you omit the structure name, system-wide <a href="#">Default Structure</a> will be searched.
----------------------	-----------------	--

<b>structureQuery</b>	<i>Optional</i>	Use this parameter to select only a part of the structure. This parameter specifies a <i>Structure Query</i> in a language similar to JQL, <a href="#">Structured JQL</a> .
-----------------------	-----------------	---



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.

### 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](#), 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\"']")
```

## 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.

## 2.7 Columns and Views

A **view** defines which **columns** are displayed in Structure and in what configuration.

The following sections will show you how to customize your columns and views to provide exactly the information you need.

## 2.7.1 Structure Columns

Structure provides a number of columns that display information about issues in the structure. You can [customize](#) the displayed columns by adding new columns, changing each column configuration, or [switching to a new view](#).

Out of the box, Structure provides the following columns:

Structure also contains [extension API](#), so the selection of available columns may be extended by a third-party plugin.

### Issue Key Column

The Issue Key column displays the issue key for issues. For other types of items it remains empty.

The screenshot shows a table with the following columns: Key, Summary, Σ Story Poi, Σ Time Sp, Progress, TP, Assignee, Status, and WSJF (Basic). The 'Key' column is highlighted with a red box and contains the following values: STMA-21, STMA-16, STMA-20, and STMA-25. The 'Summary' column contains: Team A Story 20, Folder 1, Team A Story 16, and Sub-task 4. The 'Σ Story Poi' column contains: 18, 18, 15, and 3. The 'Σ Time Sp' column contains: 18, 15, 3, and 3. The 'Progress' column contains progress bars. The 'TP' column contains icons. The 'Assignee' column contains: Unassigned, Unassigned, C. Bacca, and Unassigned. The 'Status' column contains: TO DO, TO DO, DONE, and TO DO. The 'WSJF (Basic)' column contains: 800, 143, and 800.

Key	Summary	Σ Story Poi	Σ Time Sp	Progress	TP	Assignee	Status	WSJF (Basic)
STMA-21	Team A Story 20					Unassigned	TO DO	800
	Folder 1	18						
STMA-16	Team A Story 16	18	15			Unassigned	TO DO	143
STMA-20	Sub-task 4	3				C. Bacca	DONE	
STMA-25	Bug 2					Unassigned	TO DO	800

### 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.

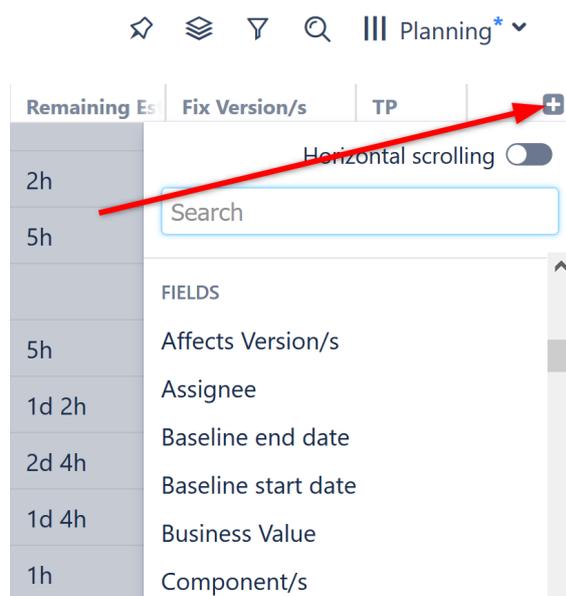
To enable compact view for the Issue Key column, open the [column options](#) and select **Compact View**.

### Summary Column

The Summary column displays the issue summary and, optionally, part of the issue description. For folders, it shows the folder name.

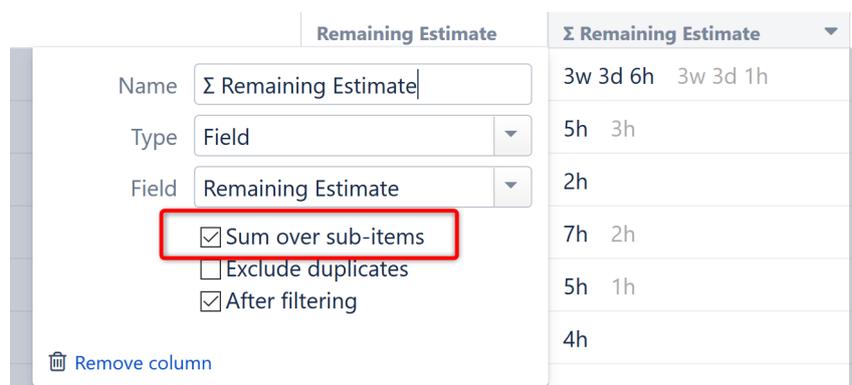
Summary can be [edited right in the structure widget](#) and it is the only field required for [creating new issues](#).





## Displaying Aggregate Values (Totals)

To display [aggregate totals](#) for a numeric or time-tracking field, check the **Sum over sub-items** box.



**i** 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](#) by double-clicking it (if the field is added to the Edit Screen in Jira).

TP	Assignee	Status	WSJF (Basic)
	Unassigned	BACKLOG	489
	Jack Brown		718
	Nah Duo	IN PROGRESS	349
	Jack Brown	IN PROGRESS	363

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.

## Icons Column

The Icons column displays icons for issue type, priority, project, reporter and assignee.

Manual ▾ ⚙️ ⌵ 🔍 ||| Basic view\* ▾

Key	Summary	TP	Status	Progress	Σ Story Point	WSJF (Basic)
STMA-16	Team A Story 16		TO DO	<div style="width: 100%;"></div>	18 15	143
STMA-20	Sub-task 4		DONE	<div style="width: 100%;"></div>	3	
STMB-44	Bug 1		TO DO	<div style="width: 100%;"></div>		409
STMA-13	Team A Story 13		TO DO	<div style="width: 100%;"></div>	5	387
STMB-45	Sub-task 2		TO DO	<div style="width: 100%;"></div>		414
STMA-25	Bug 2		TO DO	<div style="width: 100%;"></div>		400

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.

## Progress Column

The Progress column displays an aggregate issue progress, which is calculated based on values from the issue and its sub-issues.

SAFe Overview

Basic view

Key	Summary	Progress	Status	TP	WSJF (Basic)
SPF-2	SAFe Theme 2	<div style="width: 50%; background-color: green;"></div>	IN PROGRESS	213	
SPR-2	SAFe Epic 2	<div style="width: 20%; background-color: green;"></div>	BACKLOG	374	
STMA-8	Team A Story 8	<div style="width: 70%; background-color: green;"></div>	IN PROGRESS	298	
STMA-18	Sub-task	<div style="width: 100%; background-color: green;"></div>	DONE		
STMA-9	Team A Story 9	<div style="width: 30%; background-color: green;"></div>	TO DO	303	
STMA-20	Sub-task	<div style="width: 100%; background-color: green;"></div>	DONE		

Progress can be based on:

- Time tracking,
- Resolution,
- Status, or
- A custom percentage field.

You can select which type of progress column to include using the [Add Column](#) 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](#).

Progress

Name: Progress

Type: Progress

Show Percentage

ISSUE PROGRESS

Based On: Time Tracking

Apply Resolution  
If an issue has non-empty Resolution, consider progress to be 100%

Σ PROGRESS

Weight: Time Estimate

Ignore parent issue progress

[Remove column](#) [Revert changes](#)



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:

## Total Progress Calculation

When individual issue progress is calculated based on [Status](#), [Percent Field](#), or [Resolution Only](#), 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.
- **Custom Numeric Field** – Sub-issues are weighted according to a value in the specified numeric field, for example, *Story Points*. 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.

Progress	Time Spent	Remaining Estimate

Name

Type

Show Percentage

---

**ISSUE PROGRESS**

Based On

Apply Resolution

If an issue has non-empty Resolution, consider progress to be 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%.
- Otherwise, if the issue has time tracking information, the progress is calculated proportionally to this issue completion%:  $(\text{Time Spent}) / (\text{Time Spent} + \text{Remaining Estimate})$
- 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 %:  $(\text{Total Time Spent}) / (\text{Total Time Spent} + \text{Total Remaining Estimate})$ 
  - 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

Summary	Progress
▼  Top Issue	
Sub-issue 1	
▼  Sub-issue 2	
Sub-sub-issue 2.1	
Sub-sub-issue 2.2	

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%

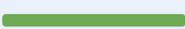
### 2. With Time Tracking Information

Summary	Progress	Time Spent	Remaining Estimate
▼  Top Issue			
Sub-issue 1		3d	1d
▼  Sub-issue 2			1d

Issue	Explanation	Progress
Sub-issue 1	It has 3 days of work logged with 1 day remaining, so its progress is $\text{time spent} / \text{total time} = 3 / (3 + 1)$ .	75%

Issue	Explanation	Progress
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 is $3 / (3 + 2)$ .	60%

### 3. More Complex Example

Summary	Progress	Time Spent	Remaining Estimate
▼  Top Issue			
 Sub-issue 1		3d	1d
▼  Sub-issue 2			1d
 Sub-sub-issue 2.1		2d	1d
 Sub-sub-issue 2.2		1d	
 Sub-issue 3			

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%
	This one has 3 days of work logged and 1 day remaining, so its progress is $3 / (3 + 1)$ .	75%

Issue	Explanation	Progress
Sub-issue 1		
Top issue	<p>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 <i>Sub-issue 3</i>, which does not have any estimates or work logged, so it gets estimated based on the average of its siblings - <i>Sub-issue 1</i> and <i>Sub-issue 2</i>.</p> <p>The progress of the top issue is calculated as follows:</p> <ul style="list-style-type: none"> <li>• The average between the total time of <i>Sub-issue 1</i> (<math>3 + 1 = 4</math> days) and the total time of <i>Sub-issue 2</i> (<math>3 + 2 = 5</math> days) is 4.5 days. So <i>sub-issue 3</i> is treated as if it has a total time of 4.5 days.</li> <li>• 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.</li> <li>• 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 <math>6 / (6 + 7.5)</math>.</li> </ul>	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.

Progress

Name

Type

Show Percentage

**ISSUE PROGRESS**

Based On

Apply Resolution

If an issue has non-empty Resolution, consider progress to be 100%

**Σ PROGRESS**

Weight

Remove column

## 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](#)

**Σ PROGRESS**

Weight

Remove column

## 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%
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 $(0 \times 2 + 100 \times 3) / (2 + 3)$ .	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 $(60 \times 5 + 0 \times 1) / (5 + 1)$ .	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.

Progress

Name: Progress

Type: Progress

Show Percentage

ISSUE PROGRESS

Based On: Status

% by Status	Status	Percentage	Action
	Open	0%	🗑️
	To Do	0%	🗑️
	In Progress	50%	🗑️
	Reopened	80%	🗑️
	Resolved	100%	🗑️
	Closed	100%	🗑️
	Done	100%	🗑️

Add status...

Remove column

## 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 it was the progress of another sub-issue.



You can specify how an issue's weight is determined in the [column configuration panel](#)

Σ PROGRESS

Weight

Remove column



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.

Summary	Status	Progress
▼  Top Issue	OPEN	<div style="width: 0%;"><div style="width: 0%;"></div></div>
⋮ ● ▼  Sub-issue 1	IN PROGRESS	<div style="width: 50%;"><div style="width: 50%;"></div></div>
 Sub-sub-issue 1.1	OPEN	<div style="width: 0%;"><div style="width: 0%;"></div></div>
 Sub-sub-issue 1.2	IN PROGRESS	<div style="width: 50%;"><div style="width: 50%;"></div></div>
✓  Sub-sub-issue 1.3	RESOLVED	<div style="width: 100%;"><div style="width: 100%;"></div></div>
✓  Sub-sub-issue 1.4	CLOSED	<div style="width: 100%;"><div style="width: 100%;"></div></div>
✓ ▼  Sub-issue 2	RESOLVED	<div style="width: 100%;"><div style="width: 100%;"></div></div>
 Sub-sub-issue 2.1	REOPENED	<div style="width: 80%;"><div style="width: 80%;"></div></div>
 Sub-sub-issue 2.2	TO DO	<div style="width: 0%;"><div style="width: 0%;"></div></div>

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 its 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.

The screenshot shows the configuration for a 'Progress' field in Jira. The field is named 'Progress' and is of type 'Progress'. The 'Show Percentage' checkbox is unchecked. Under the 'ISSUE PROGRESS' section, the 'Based On' dropdown is set to 'Custom Percent Field' and the 'Field with %' dropdown is set to 'Complete'. The 'Apply Resolution' checkbox is checked, with a note: 'If an issue has non-empty Resolution, consider progress to be 100%'. Under the 'Σ PROGRESS' section, the 'Weight' dropdown is set to 'All Sub-Issues Are Equal'. At the bottom, there are buttons for 'Remove column' and 'Revert changes'.

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.

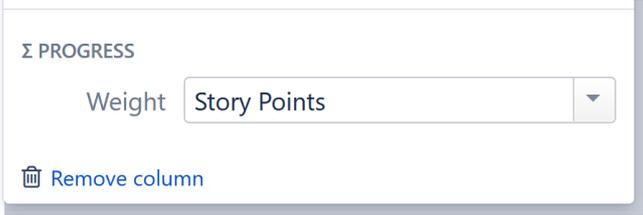
## Calculating Progress for Issue with Sub-Issues

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.

**i** You can specify how an issue's weight is determined in the [column configuration panel](#)



## 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.

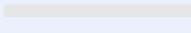
Summary	Complete	Progress
<ul style="list-style-type: none"> <li>Top Issue</li> </ul>		<div style="width: 50%;"><div style="background-color: green; height: 10px;"></div></div>
<ul style="list-style-type: none"> <li>Sub-issue 1                             <ul style="list-style-type: none"> <li>Sub-sub-issue 1.1</li> <li>Sub-sub-issue 1.2</li> <li>Sub-sub-issue 1.3</li> <li>Sub-sub-issue 1.4</li> </ul> </li> <li>Sub-issue 2</li> </ul>	50	<div style="width: 50%;"><div style="background-color: green; height: 10px;"></div></div>
<ul style="list-style-type: none"> <li>Sub-sub-issue 1.2</li> </ul>		<div style="width: 100%;"><div style="background-color: green; height: 10px;"></div></div>
<ul style="list-style-type: none"> <li>Sub-sub-issue 1.4</li> </ul>	0	<div style="width: 0%;"><div style="background-color: green; height: 10px;"></div></div>
<ul style="list-style-type: none"> <li>Sub-issue 2</li> </ul>	25	<div style="width: 25%;"><div style="background-color: green; height: 10px;"></div></div>

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, 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.

Summary	Complete	Story Points	Progress
▼  Top Issue			
▼  Sub-issue 1			
 Sub-sub-issue 1.1	50	2	
✓  Sub-sub-issue 1.2		3	
 Sub-sub-issue 1.3			
⋮ ●  Sub-sub-issue 1.4	0		
 Sub-issue 2	25	1	

Issue	Explanation	Progress
	This issue is 50% complete as specified by the custom field.	50%

Issue	Explanation	Progress
Sub-sub-issue 1.1		
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
Sub-sub-issue 1.4	This issue has a 0 "Complete" value, which means it's 0% complete.	0%
Sub-issue 1	<p>It has four sub-issues, and weight is based on story points:</p> <ul style="list-style-type: none"> <li>• 1.1 has 2 story points</li> <li>• 1.2 has 3 story points</li> <li>• 1.3 is ignored (because its progress is undefined)</li> <li>• 1.4 has no story points, so it's weight is calculated as the mean of its siblings (2 and 3 = 2.5)</li> </ul> <p>The total progress is the weighted average of 1.1, 1.2 and 1.4: <math>(50 \times 2 + 100 \times 3 + 0 \times 2.5) / (2 + 3 + 2.5)</math>.</p>	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 $(53 \times 7.5 + 25 \times 1) / (7.5 + 1)$ .	50%

## Images Column

The Images column displays small thumbnails of attached image files and allows users to view those images in a pop-up dialog.

	Key	Summary	Images	Progress
•	SP-2	Bulldoze French Alps		<div style="width: 50%;"></div>
	SP-3	Remove waste rock and s		<div style="width: 100%;"></div>

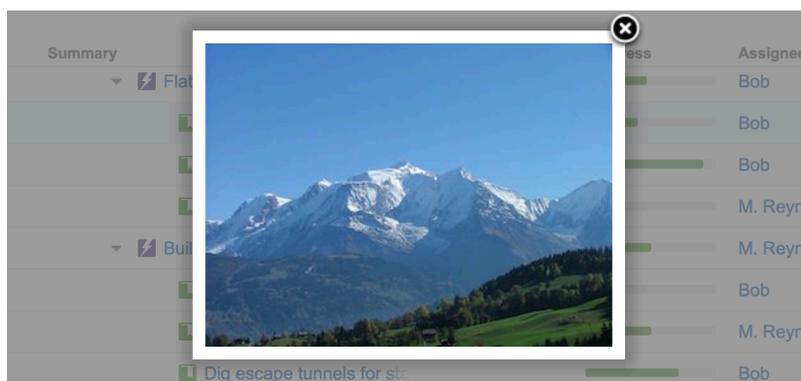
## Viewing Full-size Images

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*

## 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
<ul style="list-style-type: none"> <li>⚡ Epic 1                             <ul style="list-style-type: none"> <li>📖 Story 1</li> <li> <ul style="list-style-type: none"> <li>📖 Story 2</li> <li>🐛 Bug 1</li> </ul> </li> <li>📖 Story 3</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>4h</li> <li>5h</li> <li>2h</li> <li>1d</li> </ul>

### 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](#), 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.

🌟 🏠 🔍 🔍 ||| Work Logged\* ▾

Status	Progress	Work Logged - M. Reynold ▾
		2d 4h
		5h
		2h

Name:

Type:

Period:

Day start is calculated according to time zone America/New\_York

From:  📅

To:  📅

Logged by:

Sum over sub-items

[Remove column](#) [Revert changes](#)

### Displaying Aggregate Values

The Work Logged column can also display aggregated values, calculated as the sum of time spent over sub-issues.

Summary	Work Today	Σ Work Today
<ul style="list-style-type: none"> <li>⚡ Epic 1                             <ul style="list-style-type: none"> <li>📖 Story 1</li> <li> <ul style="list-style-type: none"> <li>📖 Story 2</li> <li>🐛 Bug 1</li> </ul> </li> <li>📖 Story 3</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>4h</li> <li>5h</li> <li>2h</li> <li>1d</li> </ul>	<ul style="list-style-type: none"> <li>2d 3h</li> <li>4h</li> <li>7h 5h</li> <li>2h</li> <li>1d</li> </ul>

To display aggregated values:

- From the Add Columns menu, select **Worked Logged...**, or
- Starting from an existing Work Logged column, open the [column configuration panel](#) 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](#).

- To view worked logged based on your own time zone, create your own instance of the Work Logged column.

## 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:

✓	<p><b>Resolved</b> flag means that the issue is in a Done status category. Such issues are considered completed and filtered out by the Unresolved <a href="#">Quick Transformation</a>.</p>
🔒	<p><b>Read-only</b> flag means that the current user does not have permission to add, remove or rearrange sub-items of the flagged issue.</p> <p>This flag is only displayed when:</p> <ul style="list-style-type: none"> <li>• The structure is configured to <a href="#">require Edit Issue permission on Parent Issue</a>,</li> <li>• The user does not have permission to <a href="#">edit this issue</a>.</li> </ul>

## 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](#) available for the issue.

Summary	TP	Status
▼ Epic 1	⚡ ⬆️	TO DO
Story 1	📌 ⬆️	IN PROGRESS
▼ Story 2	📌 ⬆️	TO DO
Bug 1	🔴 ⬆️	DONE
Story 3	📌 ⬆️	IN PROGRESS

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

Summary	TP	Status
▼ Epic 1	⚡ ⬆️	TO DO
Story 1	📌 ⬆️	
▼ Story 2	📌 ⬆️	
Bug 1	🔴 ⬆️	
Story 3	📌 ⬆️	

View Issue  
 In Progress  
 To Do  
 Done  
 Edit

⚠️ The ... button is only visible when the row is selected or when you hover over the Jira Actions column with the mouse.

### Sequential Index Column

The Sequential Index column displays a hierarchical number corresponding to an item's position within the structure.

Key	Summary	Progress	TP	Status	WSJF (Basic)	Index
STMB-40	▼ Epic 1	<div style="width: 50%;"></div>	⚡ ⬆️	TO DO	800	1
STMB-31	Story 1	<div style="width: 75%;"></div>	📌 ⬆️	IN PROGRESS	326	1.1
STMB-32	▼ Story 2	<div style="width: 50%;"></div>	📌 ⬆️	TO DO	296	1.2
STMB-43	Bug 1	<div style="width: 100%;"></div>	🔴 ⬆️	DONE		1.2.1
STMA-35	Bug 2	<div style="width: 0%;"></div>	🔴 ⬆️	TO DO	800	1.2.2
STMB-41	▼ Epic 2	<div style="width: 50%;"></div>	⚡ ⬆️	TO DO	144	2
STMB-37	▼ Story 4	<div style="width: 75%;"></div>	📌 ⬆️	IN PROGRESS	326	2.1
STMA-36	Task	<div style="width: 0%;"></div>	📌 ⬆️	TO DO	800	2.1.1
STMB-39	Story 6	<div style="width: 50%;"></div>	📌 ⬆️	TO DO	338	2.2

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 ignores [Filter Transformations](#) – even if you see only a part of a structure, the numbers will still show the position of the item in the unfiltered structure. [Filter generators](#) will change the sequential index, because they change the underlying structure.

## 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 ne	OPEN	should be closed. see the report
SP-6	🔍 ⬆️	▶️ 🔍 Build access road - We need an eight-l	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 ne	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](#) 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.

## 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.

Manual ▾

☆ ☰ ▾ 🔍 III Estimates\* ▾

Key	Summary	Remaining Estimate	Σ Remaining Estimate
STMA-16	Team A Story 16	3w 3d 1h	3w 3d 6h 3w 3d 1h
✓ STMA-20	Sub-task 4	3h	5h 3h
STMB-44	Bug 1	2h	2h
STMA-13	Team A Story 13	2h	7h 2h
STMB-45	Sub-task 2	1h	5h 1h
STMA-25	Bug 2	4h	4h

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](#).

## Sum over sub-items

All aggregate columns have the Sum over sub-items box checked.

Remaining Estimate	Σ Remaining Estimate
3w 3d 6h	3w 3d 1h
5h	3h
2h	
7h	2h
5h	1h
4h	

Name

Type

Field

Sum over sub-items

Exclude duplicates

After filtering

[Remove column](#)

This is what makes them aggregate columns. Unchecking this box will change the column from an aggregate column to its 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](#) 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.

## 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.

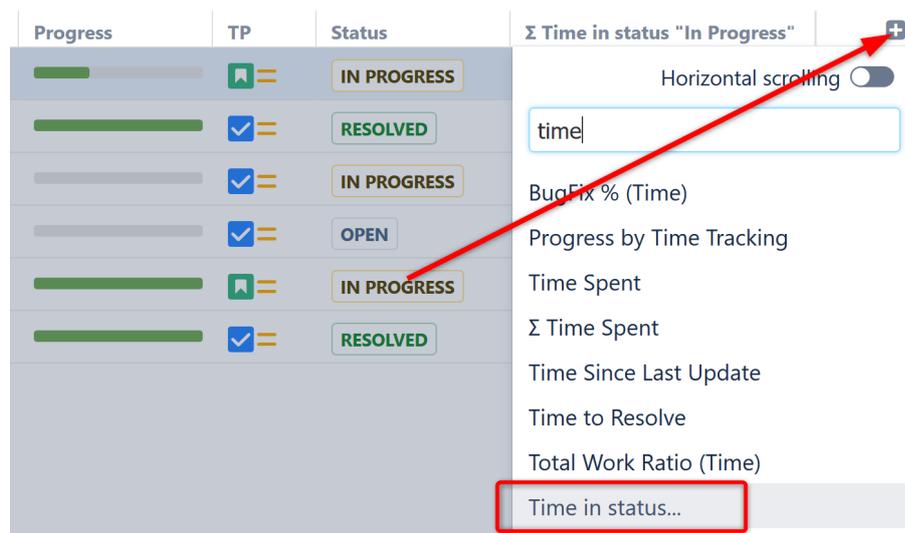
Simple Structure ▾ ☆ ⚙️ ▾ 🔍 ||| Basic view ▾

Key	Summary	Status	Time in Status "Open"	Time in Status "Open" and "In Progress"	Time in Status "Open" transitioned to "In Progress"
DT-27	Story 1	IN PROGRESS	3h 45m	4w 6d 49m	3h 45m
DT-30	Task 1.1	RESOLVED	3h 44m	3h 49m	3h 44m
DT-31	Task 1.2	IN PROGRESS	3h 44m	4w 6d 48m	3h 44m
DT-33	Task 1.3	OPEN	5d 20h 11m	5d 20h 56m	0m
SCP-1	Project B - Story 1	TO DO	0m	0m	0m
SCP-2	Project B - Task 1.1	DONE	0m	0m	0m

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](#) 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](#) 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](#) (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).

Time in Status	Progress	TP	Status	Status
3h 45m	Name			
3h 44m	Type			
3h 44m	Statuses			
5d 20h 11m	Conditions			
0m	Options			
0m				

**Name**

**Type**

**Statuses**

**Conditions**  Returns only ?  
 Ignore the first time the issue is in the selected group of statuses.

Transition filter  
 Only count time in a status if the issue then moved directly to a target status.

**Options**  Sum over sub-items

Work time  
 When checked, the value will be interpreted according to Jira's "work hours per day" settings.

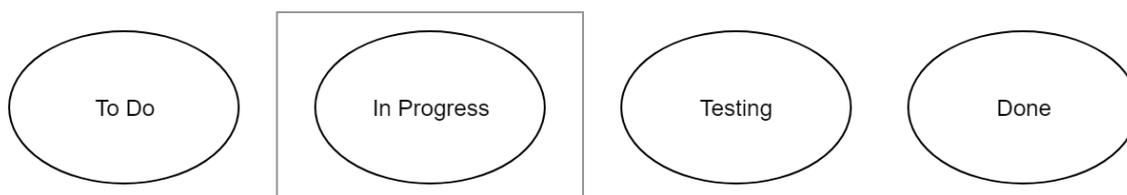
[Remove column](#)

## 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.

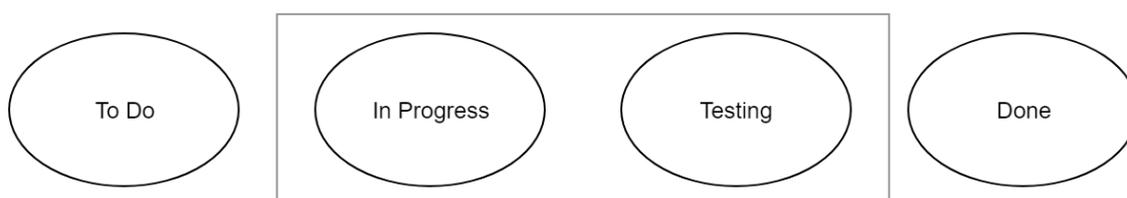


In this configuration, Structure would ignore the first time the issue was in the **In Progress** status, and only track time if it returned to the **In Progress** status after moving to Testing, Done, or back to To Do.

### 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**.

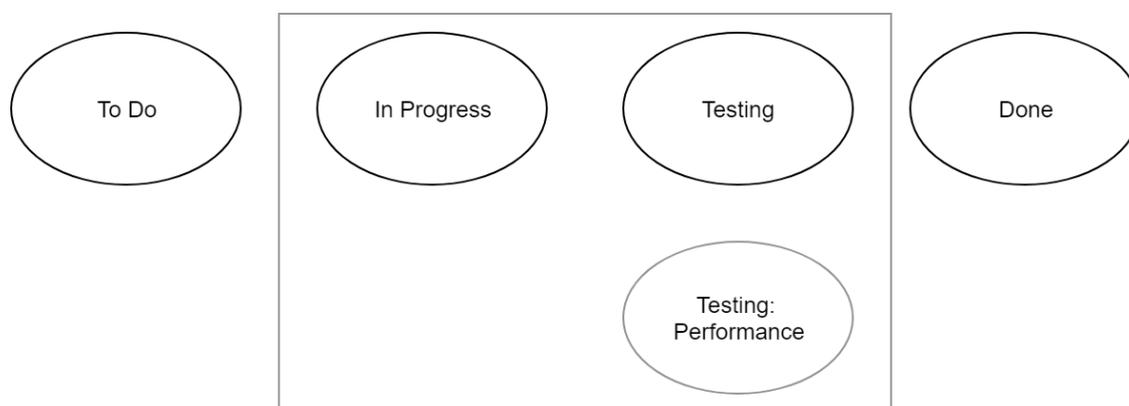
To do this, you could add both **In Progress** and **Testing** to your selected Statuses, and continue to select the **Returns only** box.



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.

### 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.

Simple Structure ▾

Basic view\*

Key	Summary	Progress	TP	Status	Status Category
DT-28	Story 1	<div style="width: 50%;"></div>		IN PROGRESS	IN PROGRESS
DT-30	Task 1.1	<div style="width: 100%;"></div>		RESOLVED	DONE
DT-31	Task 1.2	<div style="width: 100%;"></div>		IN PROGRESS	IN PROGRESS
DT-36	Task 1.3	<div style="width: 100%;"></div>		OPEN	TO DO
SCP-1	Project B - Story 1	<div style="width: 50%;"></div>		TO DO	TO DO
SCP-2	Project B - Task 1.1	<div style="width: 100%;"></div>		DONE	DONE

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](#).

Progress	TP	Status	Status category	Formula
<div style="width: 50%;"></div>		IN PROGRESS	IN PROGRESS	
<div style="width: 100%;"></div>		RESOLVED	DONE	1
<div style="width: 100%;"></div>		IN PROGRESS	IN PROGRESS	
<div style="width: 100%;"></div>		OPEN	TO DO	
<div style="width: 50%;"></div>		TO DO	TO DO	
<div style="width: 100%;"></div>		DONE	DONE	1

Name

Type

Formula

Variables  StatusCategory  
One variable used. Click the variable to define it.

Options  Sum over sub-items

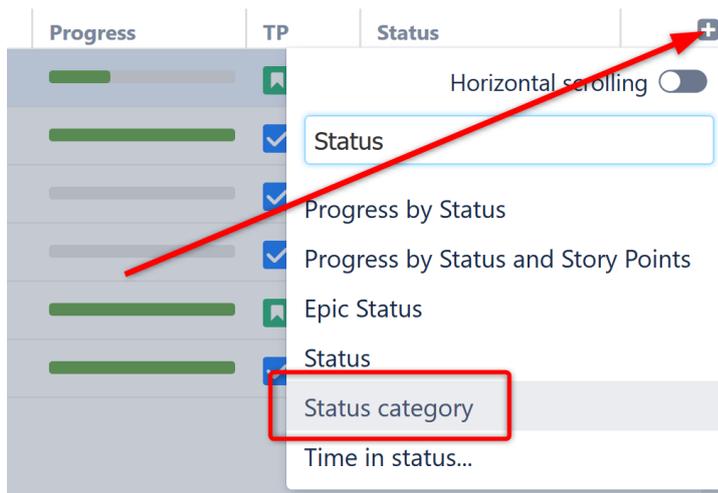
Format

[Remove column](#) [Revert changes](#)

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**.



## 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.

Counting Sub-Items and Leaves ▾ ☆ ⚙️ 🔍 Basic view\* ▾

Key	Summary	Count Sub-Items
INI-1	Core Products	21
INI-3	Structure	18
	Roadmap Features	14
STR-3	Formulas	6
STR-6	As a formula author, I want to edit large formulas in the setting panel	2
STR-7	UX Design: larger dialog for editing formula	
STR-8	Implement new UX	
STR-15	As a formula author, I want to be able to use JLQ queries	1
STR-16	Design syntax	

Structure with Count Sub-Items column

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](#).

## 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](#) or [folders](#) and you need a count of just the issues themselves.

Bottom-Up Automation ▾ ☆ ☰ 🔍 Compact\* ▾

Summary	Count Sub-Items	Count Leaves
▾ HP Sprint 4	13	9
▾ 👤 Albert	1	1
📄 Additional Story 2		
▾ 👤 Anna M.	2	2
📄 Additional Story 3		
📄 Additional Story 4		

Structure with Count Leaves column

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](#)

- ✔ You can toggle between Count Sub-Items and Count Leaves. Open the column configuration, click the settings icon and check/uncheck **Leaves Only**.

Count Leaves
11
10
9
4
2

Count Sub-Items column option for Leaves Only

## 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.

SAFe Overview

Transition Dates

Key	Summary	Status	First Transition to In Progress	Latest Transition to Done
SPF-2	SAFe Theme 2	IN PROGRESS	13/Apr/18 10:29 AM	
SPR-2	SAFe Epic 2	BACKLOG		
STMA-8	Team A Story 8	IN PROGRESS	21/Dec/18 1:23 PM	
STMA-18	Sub-task 2	DONE	13/Apr/18 8:56 AM	24/Dec/18 3:45 PM
STMA-9	Team A Story 9	TO DO		
STMA-20	Sub-task 4	DONE		21/Dec/18 1:21 PM

### Transition Dates in Formulas

These Transition Date values can also be used in formulas by assigning a variable to the appropriate Transition Date attribute.

First Transition to In Progress | Latest Transition to Done

Name: Time from In Progress to Done

Type: Formula

Formula: `If (Status = "Done"; LatestTransDone - FirstTransInProgress)`

Variables: < Back to Variable List

LatestTransDone

Select attribute...

Options: Recently Used

Format: Latest Transition to Done

Used in Columns

First Transition to In Progress

Latest Transition to Done

Remove column

## Query Match Column

The Query Match column allows you to easily identify issues that match a specified text, JQL or S-JQL query. Items that match the query are noted by a check mark; those that do not are left blank.

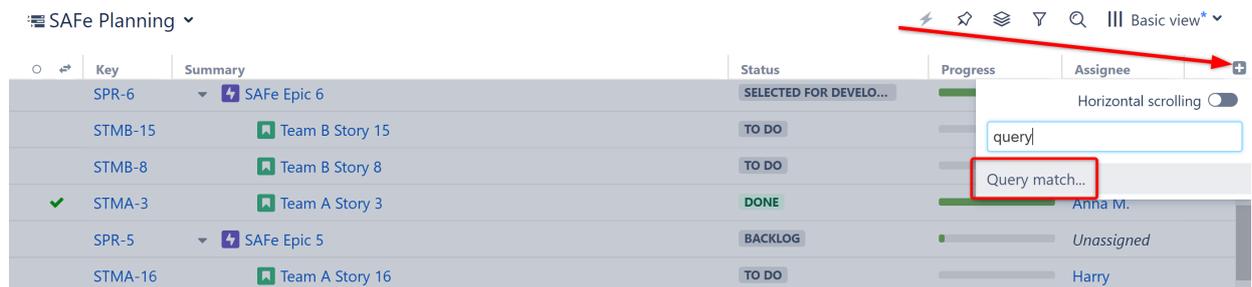


Key	Summary	Status	Progress	Assignee	JQL query
STMB-7	Team B Story 7	IN PROGRESS	<div style="width: 50%;"></div>	M. Reynolds	✓
STMB-6	Team B Story 6	TO DO	<div style="width: 0%;"></div>	Anna M.	
STMB-4	Team B Story 4	TO DO	<div style="width: 0%;"></div>	Albert	
SPR-12	SAFe Epic 12	BACKLOG	<div style="width: 20%;"></div>	Unassigned	
STMB-11	Team B Story 11	IN PROGRESS	<div style="width: 60%;"></div>	Harry	✓
STMB-5	Team B Story 5	IN PROGRESS	<div style="width: 70%;"></div>	Claire T.	✓
STMA-2	Team A Story 2	IN PROGRESS	<div style="width: 30%;"></div>	Anna M.	✓

Query Match column

## 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...**



Key	Summary	Status	Progress	Assignee
SPR-6	SAFe Epic 6	SELECTED FOR DEVELO...	<div style="width: 100%;"></div>	
STMB-15	Team B Story 15	TO DO	<div style="width: 0%;"></div>	
STMB-8	Team B Story 8	TO DO	<div style="width: 0%;"></div>	
STMA-3	Team A Story 3	DONE	<div style="width: 100%;"></div>	Anna M.
SPR-5	SAFe Epic 5	BACKLOG	<div style="width: 0%;"></div>	Unassigned
STMA-16	Team A Story 16	TO DO	<div style="width: 0%;"></div>	Harry

Add a Query Match column

In the Column Configuration panel, select the type of query (text, JQL or S-JQL) and enter your query. The name will update automatically based on your query, but you can also enter a custom name.

Key	Summary	Status	Progress	Assignee	Creator
SPR-6	SAFe Epic 6	SELECTED FOR EDIT			
STMB-15	Team B Story 15	TO DO			
STMB-8	Team B Story 8	TO DO			
STMA-3	Team A Story 3	DONE			
SPR-5	SAFe Epic 5	BACKLOG			
STMA-16	Team A Story 16	TO DO			
STMA-5	Team A Story 5	DONE			
STMA-4	Team A Story 4	DONE		Harry	
SPR-10	SAFe Epic 10	BACKLOG		Unassigned	

### Query Match settings

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 [Query Match as Variables](#) to learn more.

### 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 **Sum over sub-items**. This allows you to aggregate your results up the hierarchy.

Key	Summary	Status	Progress	Assignee	Σ JQL query
SPR-9	SAFe Epic 9	SELECTED FOR EDIT			1
STMB-14	Team B Story 14	TO DO			
STMB-9	Team B Story 9	TO DO			
STMB-7	Team B Story 7	IN PROGRESS			1
STMB-6	Team B Story 6	TO DO			
STMB-4	Team B Story 4	TO DO			
SPR-12	SAFe Epic 12	BACKLOG			3
STMB-11	Team B Story 11	IN PROGRESS			1
STMB-5	Team B Story 5	IN PROGRESS			1
STMA-2	Team A Story 2	IN PROGRESS		Anna M.	1
SPR-3	SAFe Epic 3	IN PROGRESS		Unassigned	5 1
STMB-10	Team B Story 10	IN PROGRESS		Nah Duo	2 1

### Sum over sub-items with Match Query

In this example, Safe Epic 3 has 5 items (including itself) that were created by Admin and are in progress.

## Aggregation Options

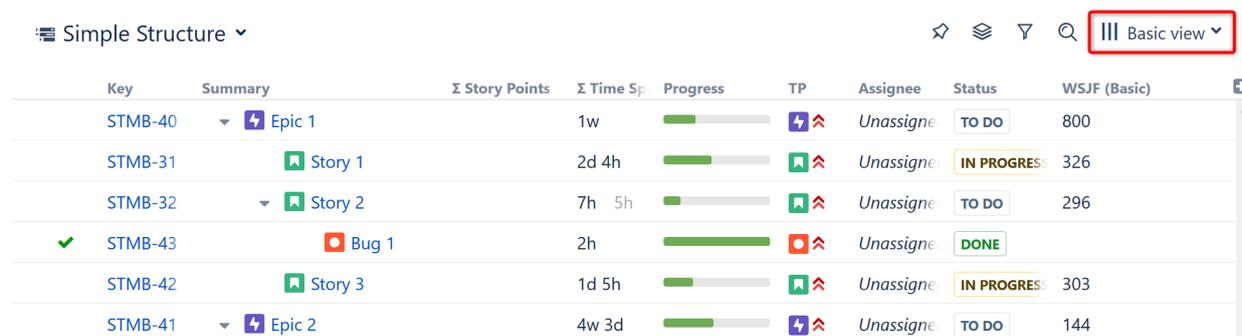
When **Sum over sub-items** is selected, you have a couple of options:

- **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.

### 2.7.2 Configuring View

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.



Simple Structure ▾ ⚙️ 🔍 📄 Basic view ▾

Key	Summary	Σ Story Points	Σ Time Sp	Progress	TP	Assignee	Status	WSJF (Basic)
STMB-40	▾ Epic 1		1w	<div style="width: 50%;"></div>	⚡ ⬆	Unassigne	TO DO	800
STMB-31	▢ Story 1		2d 4h	<div style="width: 75%;"></div>	▢ ⬆	Unassigne	IN PROGRESS	326
STMB-32	▾ Story 2		7h 5h	<div style="width: 30%;"></div>	▢ ⬆	Unassigne	TO DO	296
✓ STMB-43	🔴 Bug 1		2h	<div style="width: 100%;"></div>	🔴 ⬆	Unassigne	DONE	
STMB-42	▢ Story 3		1d 5h	<div style="width: 60%;"></div>	▢ ⬆	Unassigne	IN PROGRESS	303
STMB-41	▾ Epic 2		4w 3d	<div style="width: 40%;"></div>	⚡ ⬆	Unassigne	TO DO	144

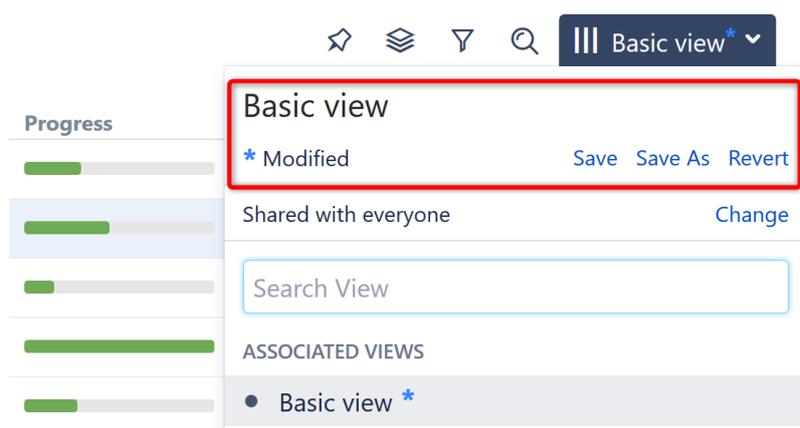


When viewing a structure from other locations in Jira, due to space limitations the view's name may not be displayed. To identify the current view, simply hover over the Views icon (the 3 vertical lines).

## Changing the View

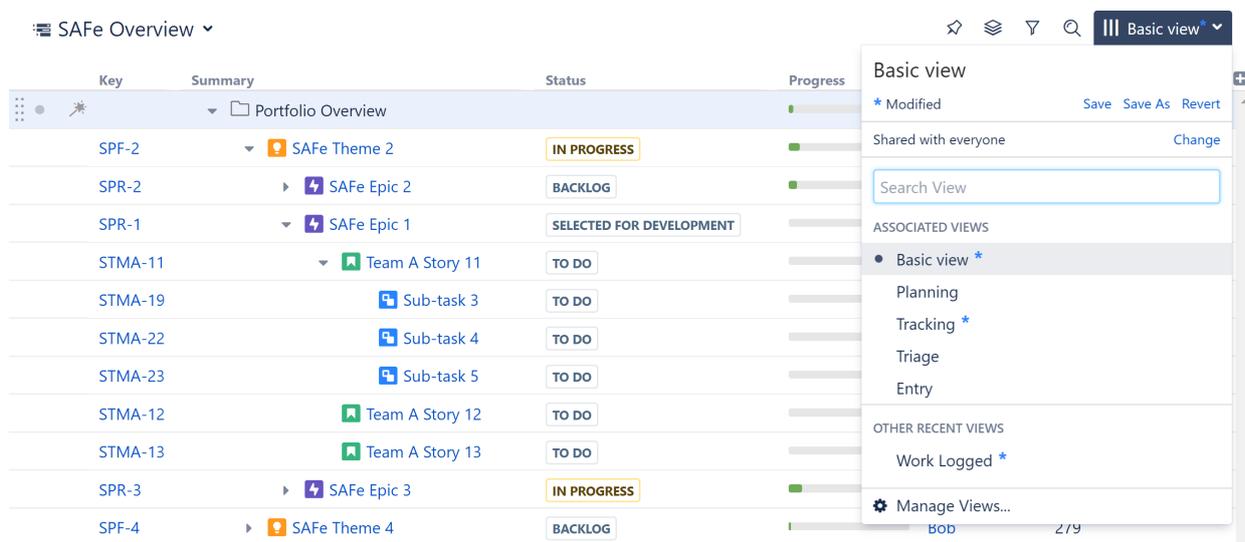
To change which columns are displayed or the order in which they appear, you can [select a new view](#) or manually [add, remove or rearrange columns](#).

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](#).



## 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](#). 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. Search View. Search looks for any views that match the entered name, not only those in this list.
5. Associated Views list. This list can be customized for each structure by the structure owner or anyone who has [Control access level](#) for the structure – see [Customizing View Settings](#).
6. List of views you have recently used (excluding the views shown in the section above).
7. Manage Views link, which opens the [View Management](#) 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.

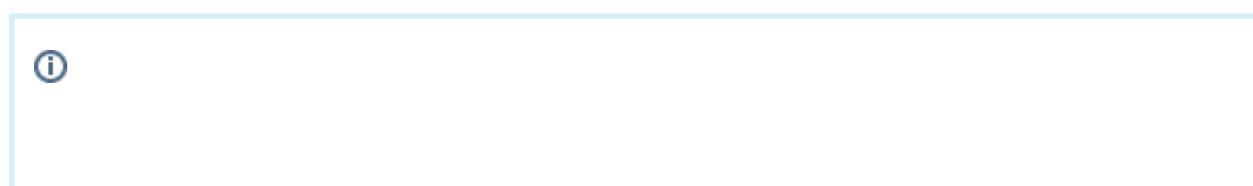
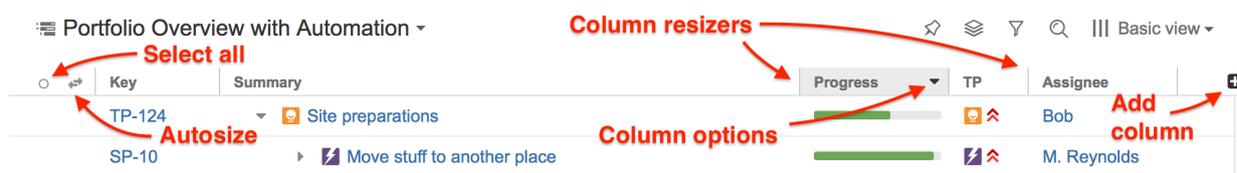
## Switching View with Keyboard

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.

## 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.

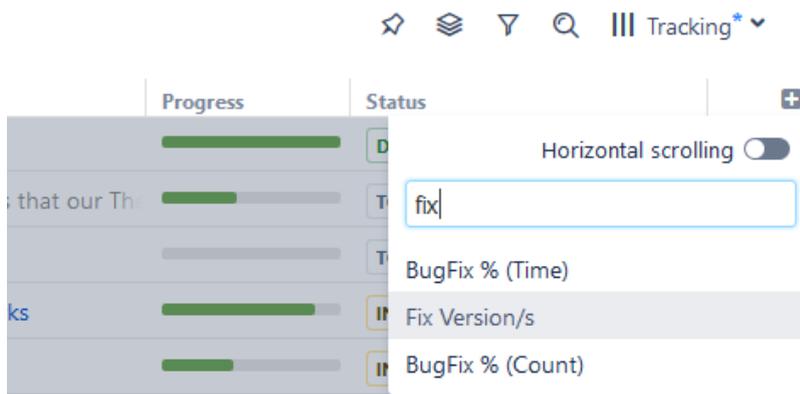


When you add, configure, remove or rearrange columns, you make adjustments to the *View* that's being used to display the structure. An adjusted view is marked with a blue asterisk ( \* ). The adjustments are stored in your browser and affect only yourself – to make the changes permanent and available to others, you need to [save the view or create a new view](#).

## 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.



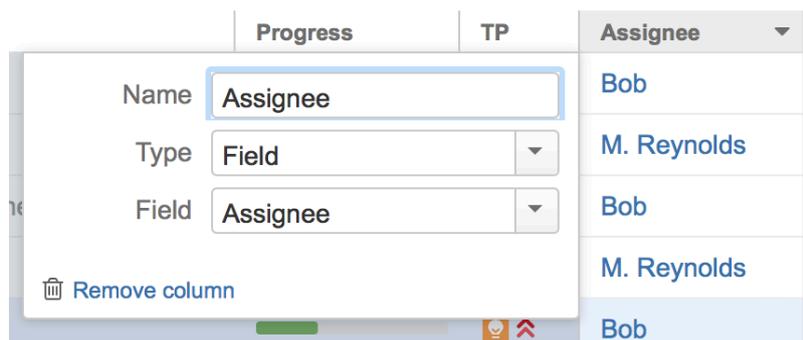
To abort adding a new column, hit **Escape**.



Use keyboard shortcut **TT** to quickly open the Add Column dialog (hit "t" twice).

## 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](#). For example, the [Field](#) 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.

## 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.

**i** You cannot remove the [Summary Column](#), [Flags Column](#) or [Jira Actions Column](#).

## 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.

## 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](#).



Column widths are not part of the *View* and are not saved on the server or shared.

## Horizontal Scrolling

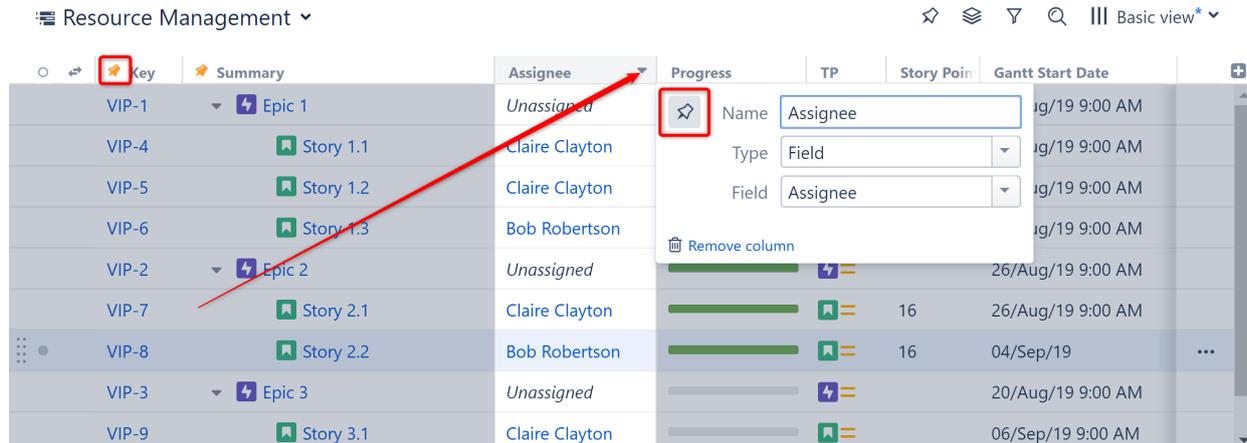
Structure automatically adjusts column sizes to fit every selected column into the panel. While you can easily change the size of each column, by default you are still bound by the size of your Structure panel. This can make it difficult to view several columns at once or columns with a great deal of 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 Add Column icon “+”
- Click the **Horizontal scrolling** toggle.





### Pinning columns

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 its 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.

**i** The Summary column cannot be unpinned.

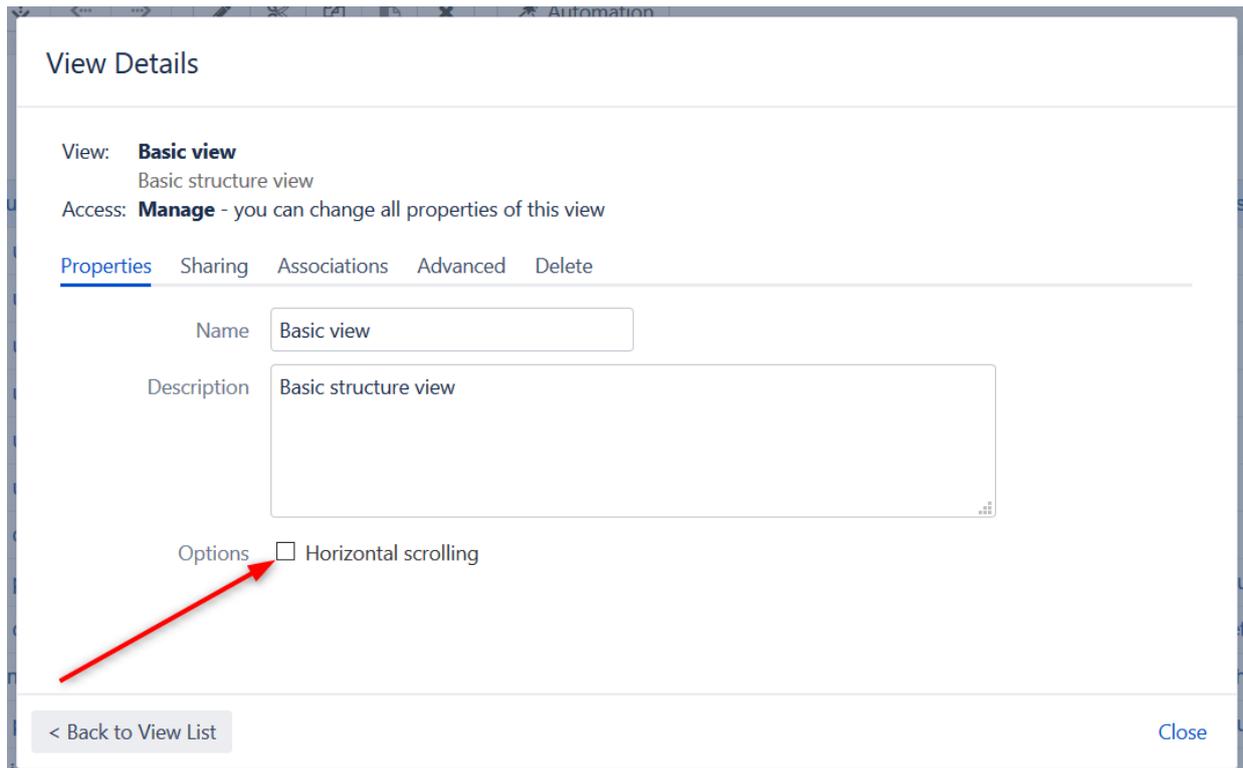
## Saving Horizontal Scrolling Settings

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**



Making horizontal scrolling the default for a view

**i** 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.

## Saving and Sharing Views

When you [add, remove or rearrange columns](#), 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.

Simple Structure ▾ ⚙️ ⌵ 🔍 ||| Basic view\* ▾

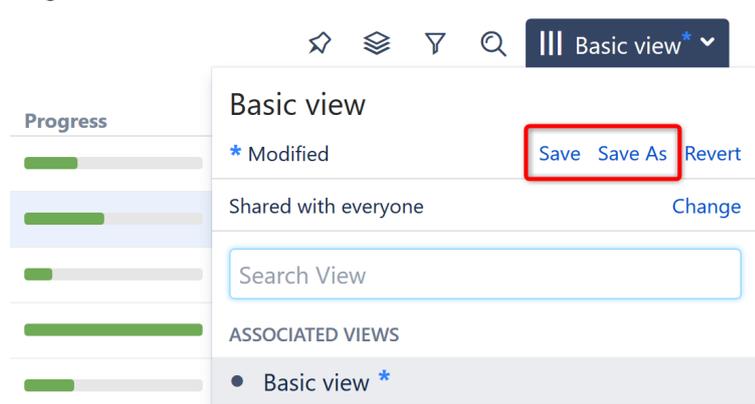
Key	Summary	Σ Time Spent	Progress	TP	Status	Original Estimate
STMB-40	▾ Epic 1	1w	<div style="width: 50%;"></div>	⚡ ⬆	TO DO	5w
STMB-31	● Story 1	2d 4h	<div style="width: 75%;"></div>	⬆ ⬆	IN PROGRESS	...
STMB-32	▾ Story 2	7h 5h	<div style="width: 25%;"></div>	⬆ ⬆	TO DO	1w
✓ STMB-43	🔴 Bug 1	2h	<div style="width: 100%;"></div>	🔴 ⬆	DONE	
STMB-42	● Story 3	1d 5h	<div style="width: 30%;"></div>	⬆ ⬆	IN PROGRESS	
STMB-41	▾ Epic 2	4w 3d	<div style="width: 40%;"></div>	⚡ ⬆	TO DO	5w
STMB-37	● Story 4	1w 2d	<div style="width: 60%;"></div>	⬆ ⬆	IN PROGRESS	2w

**i**

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.

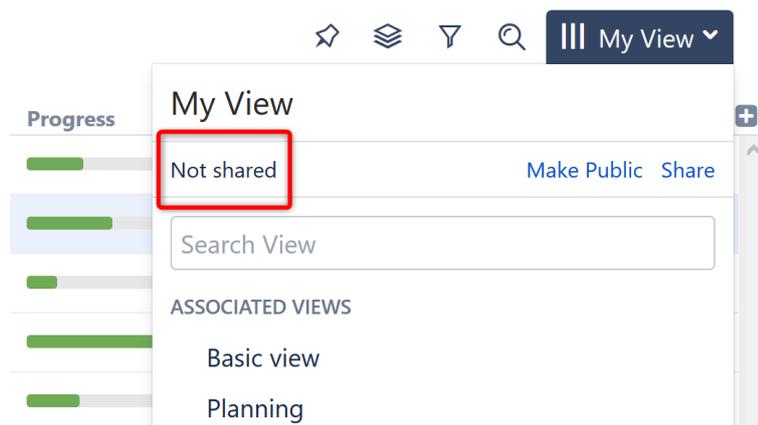


**i** To save changes to an existing view, you need to have *Update* access level for that view (see [View Sharing and Permissions](#)). If you do not have permissions to change the view, you can still create a new view based on your modifications with the **Save As** link.

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.



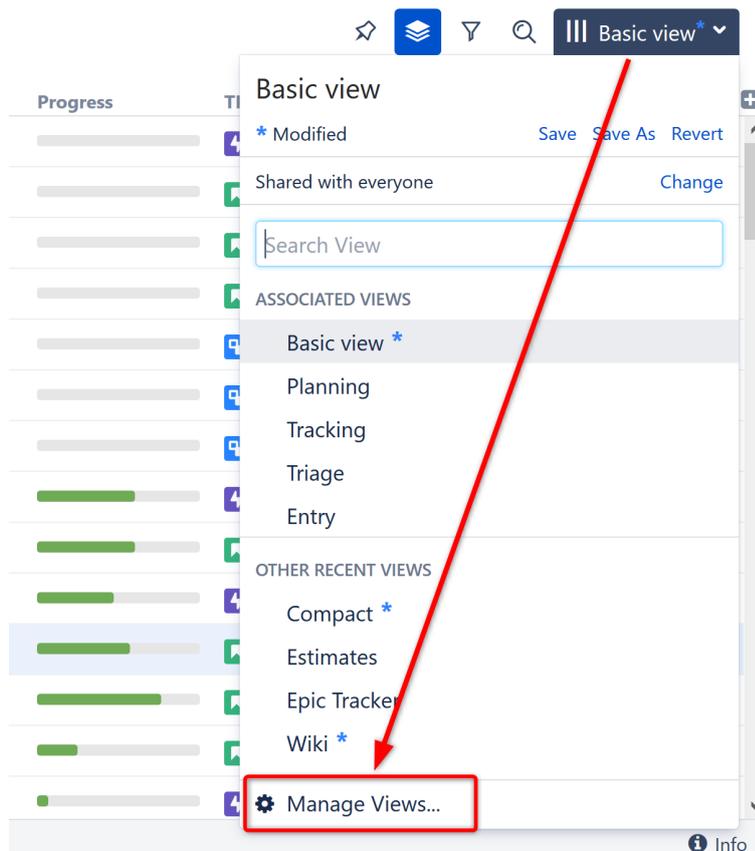
To share a view with other people, open the Views menu and:

- 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](#) for details.

### 2.7.3 Managing Views

In Structure, a **view** defines which columns are displayed and in what configuration. In addition to the preinstalled views, Structure users can also create and share their own views.

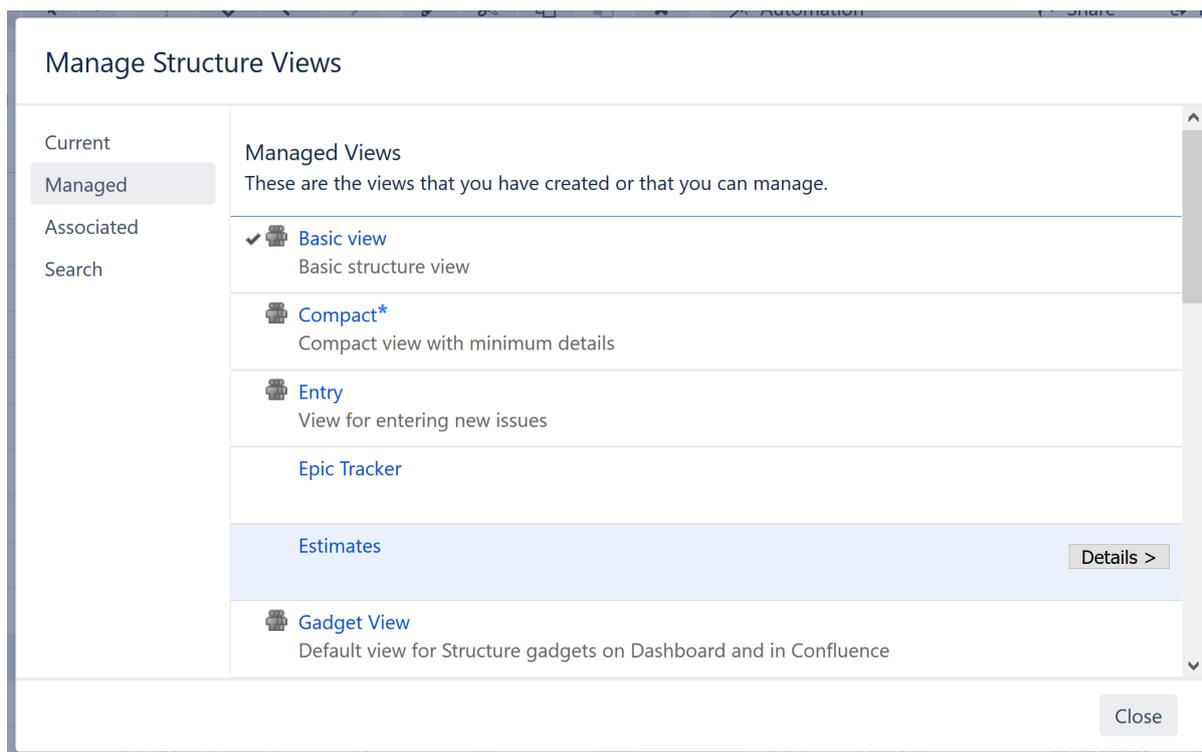
You can find, select and save views from the [Views menu](#). For additional operations, or to browse all available views, open the Views menu and select **Manage Views**:



The following sections cover specific elements of view management:

## 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](#); 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](#).

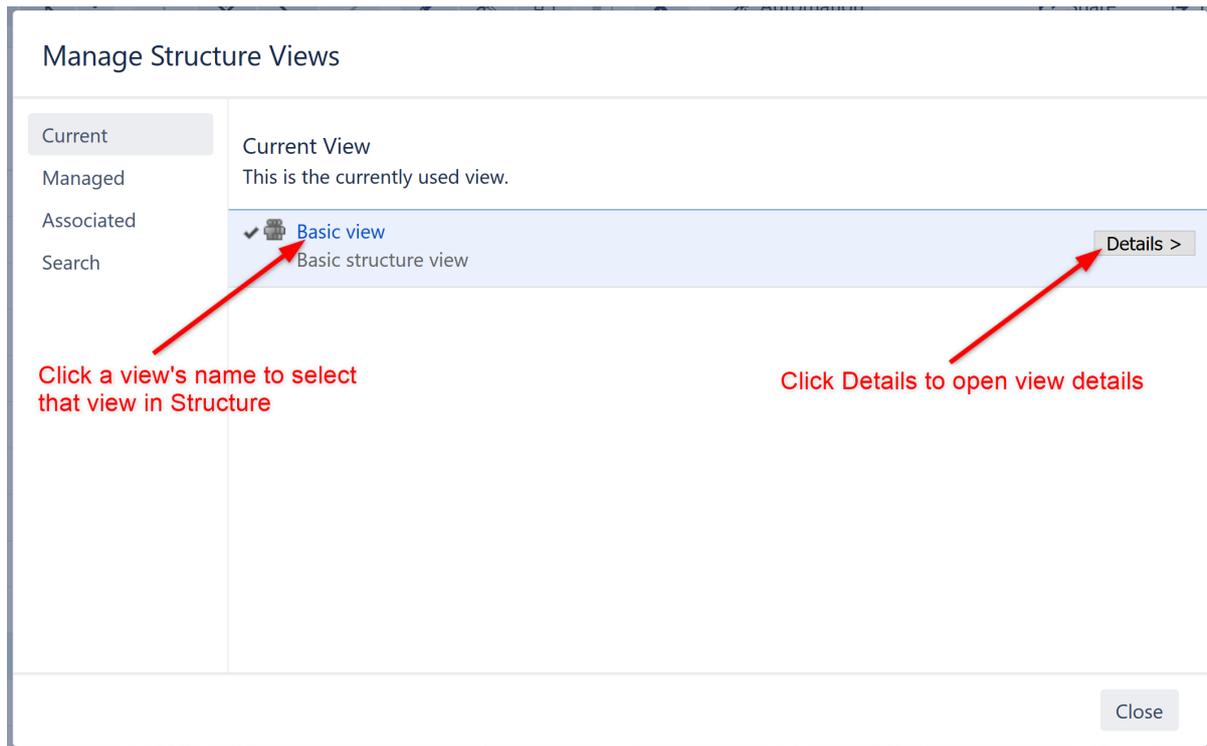


To find a particular view, select one of the following tabs:

- **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](#) 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.

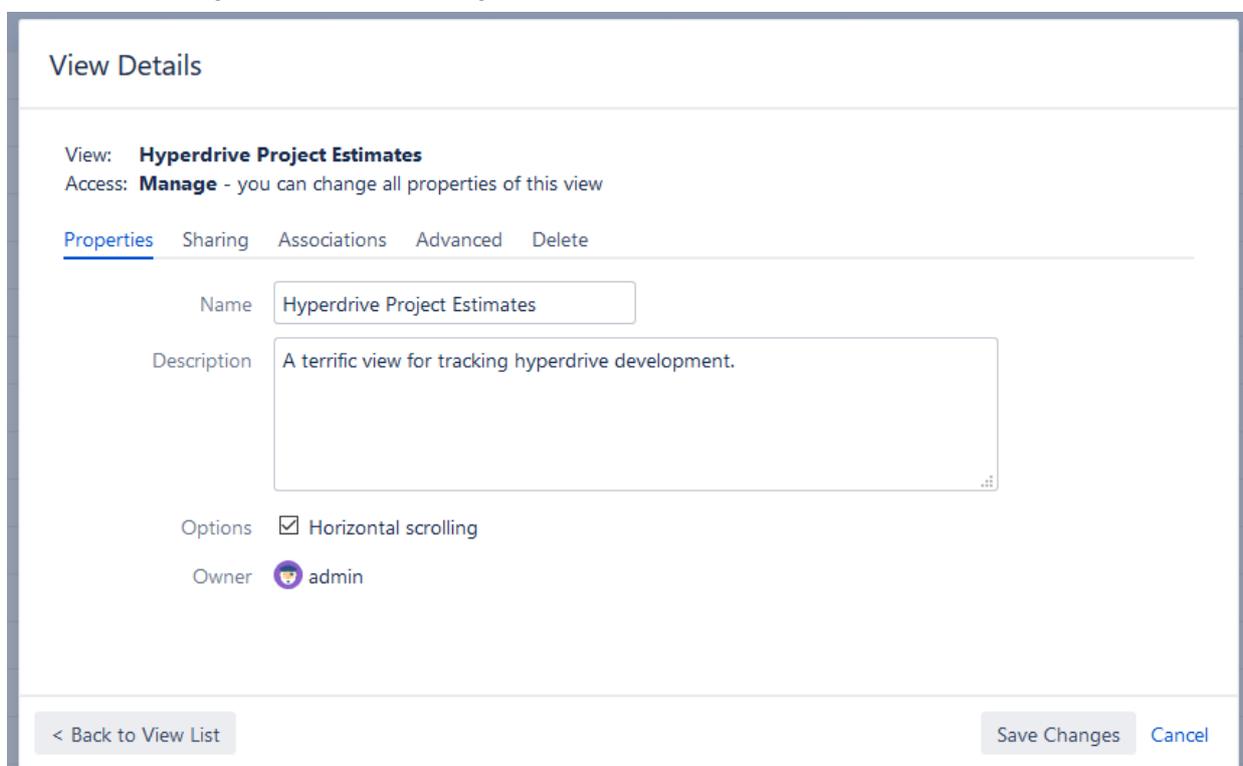
When you have located a specific view, you can click its name to switch to that view. The view will also appear in the **Other Recent Views** section of the [Views menu](#).

To see and edit View details, click the **Details** button that appears when you move the mouse pointer over the view record.



## Changing View Settings

When you have [located a view](#) 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](#) is enabled by default.
- **Sharing** - lets you view and modify sharing permissions for the view – see [View Sharing and Permissions](#).
- **Associations** - shows the structures which are associated with the view (have this view in their Views drop-down). See [Associating Views with Structures](#).
- **Advanced** - shows additional technical information about the view.
- **Delete** - lets you [delete this view](#).



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](#) page.

## 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:

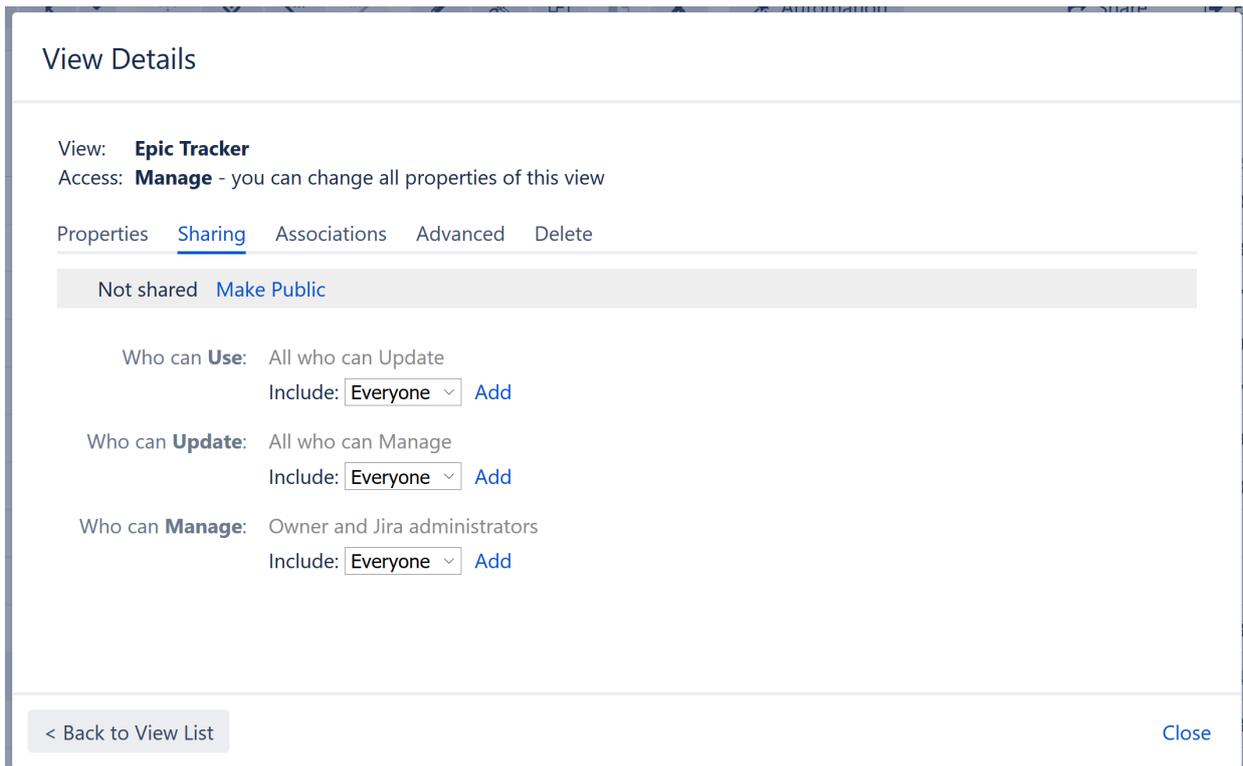
<b>None</b>	The view is not visible nor usable by the user.
<b>Use</b>	Read-only access: the user can use the view, but cannot modify it.
<b>Update</b>	The user can use the view, and also save view adjustments as the new version of the view. The user cannot modify view name or sharing permissions.
<b>Manage</b>	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.

**i** People who have only **Use** permission for a view still **can add, remove or rearrange columns**, 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)

**i** 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](#) – 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.

## 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](#) when that structure is used.



Jira administrator can also specify [global default view settings](#), 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](#) and locate the view to associate. Click **View Details** and select the **Associations** tab.

### View Details

View: **Epic Tracker**  
 Access: **Manage** - you can change all properties of this view

Properties   Sharing   Associations   Advanced   Delete

A structure is "**associated**" with this view, if the view is offered in the drop-down menu for this structure, according to the structure's **View Settings**. View settings are managed on the **Manage Structure** page. [Refresh Associations](#)

Structures: This view is not associated with any structure.  
[Add to the current structure's views...](#) <sup>Ⓔ</sup>

[< Back to View List](#) [Close](#)

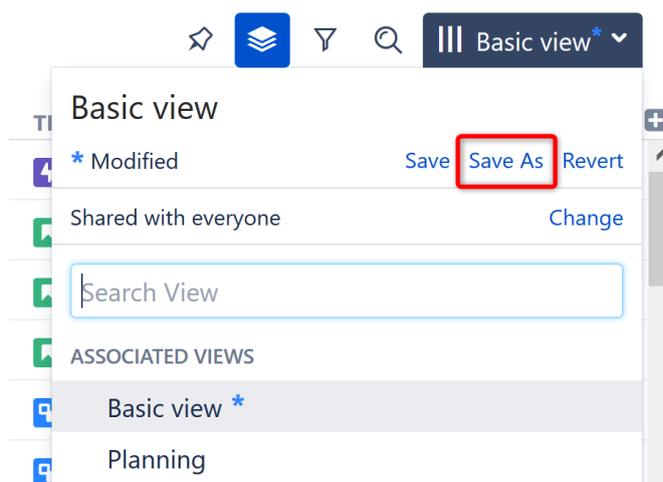
To associate the view with the current structure, click the **Add to the current structure's views...** link. If you have Manage access for the structure, this will open the [View Settings](#) for the structure and add the current view. Scroll to the bottom of the page and click **Apply**.

**i** 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.

## 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](#) or [Manage Views dialog](#) 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](#) link to create a new view.
4. Use the **Manage Views** dialog to review the new view's description and sharing permissions.



## Deleting a View

To delete a view:

1. Open the [Manage Views](#) dialog.
2. [Locate the view](#) 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](#) 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.

### 2.7.4 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.

☰ SAFe Overview ▾

Key	TP	Summary	Progress
✱		▾ Portfolio Overview	<div style="width: 100%; height: 10px; background: linear-gradient(to right, green 10%, grey 10%);"></div>
SPF-2	🔥↑	▾ 🔥 SAFe Theme 2	<div style="width: 100%; height: 10px; background: linear-gradient(to right, green 10%, grey 10%);"></div>
SPR-2	⚡↑	▾ ⚡ SAFe Epic 2	<div style="width: 100%; height: 10px; background: linear-gradient(to right, grey 10%, grey 10%);"></div>
STMA-8	👤↑	▾ 👤 Team A Story 8	<div style="width: 100%; height: 10px; background: linear-gradient(to right, grey 10%, grey 10%);"></div>
STMA-17	📄↑	📄 Sub-task 1	<div style="width: 100%; height: 10px; background: linear-gradient(to right, grey 10%, grey 10%);"></div>
STMA-18	📄↑		<div style="width: 100%; height: 10px; background: linear-gradient(to right, grey 10%, grey 10%);"></div>
⋮ ● 8	👤↑	▸ 👤 Team A Story 9	<div style="width: 100%; height: 10px; background: linear-gradient(to right, grey 10%, grey 10%);"></div>

✕ ⋮

This sub-task should be done first. It is a critical piece of this project, as many other tasks relay on this one. Seriously, don't delay!

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.

### 2.7.5 Two-Panel Mode

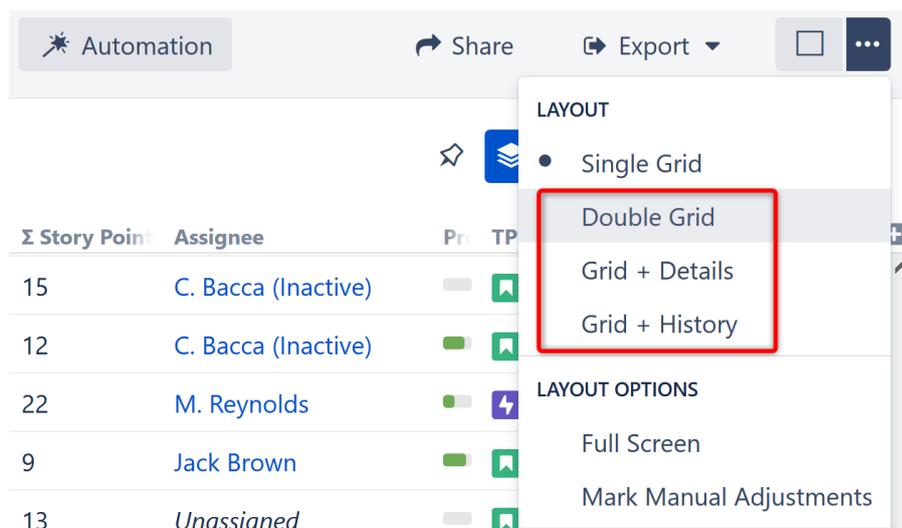
When working with a structure on the Structure Board, you can switch to the two-panel mode to take full advantage of the screen space.

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](#) panel - as you click an issue in the structure, you can see/edit the issue details in the panel on the right
- [History](#) - 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

### Viewing the Secondary Panel

To open the Secondary panel, use the **Toggle Panels** menu in the [Main Structure Toolbar](#).



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](#) for the currently-selected issue.
- **Grid + History** opens [Structure History](#).
- If you have additional add-ons that utilize the secondary panel, their options will be displayed below these.

## Resizing the Secondary Panel

You can divide the horizontal space between the secondary panel and the main panel by dragging the separating border.

## 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.

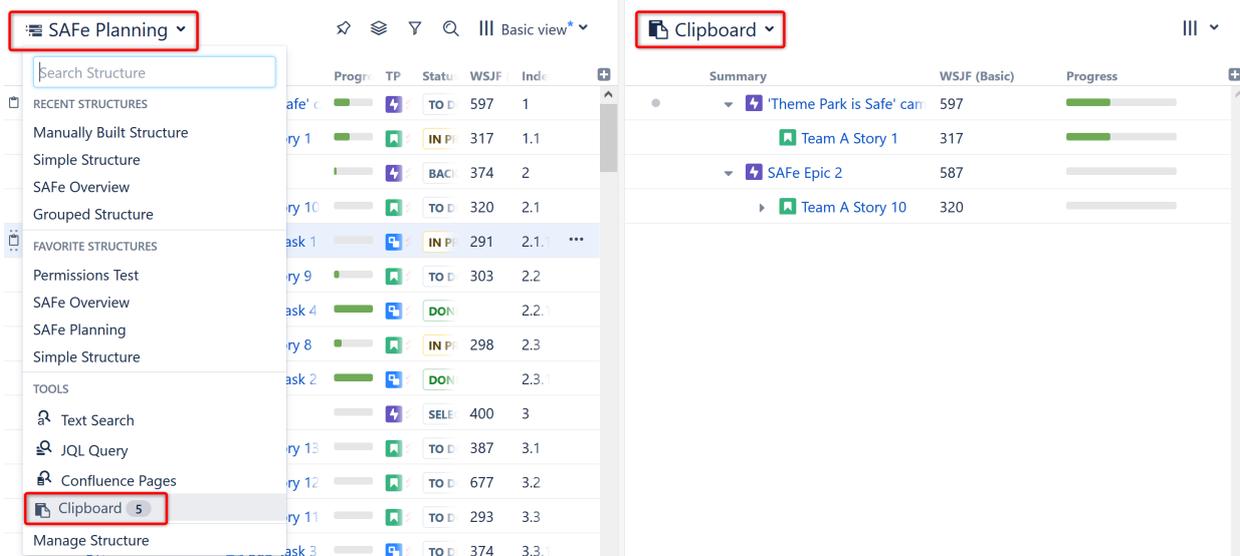


This is especially useful when you need to make sure the structure in the main panel has all the issues you've found using search in the secondary panel.

## 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.

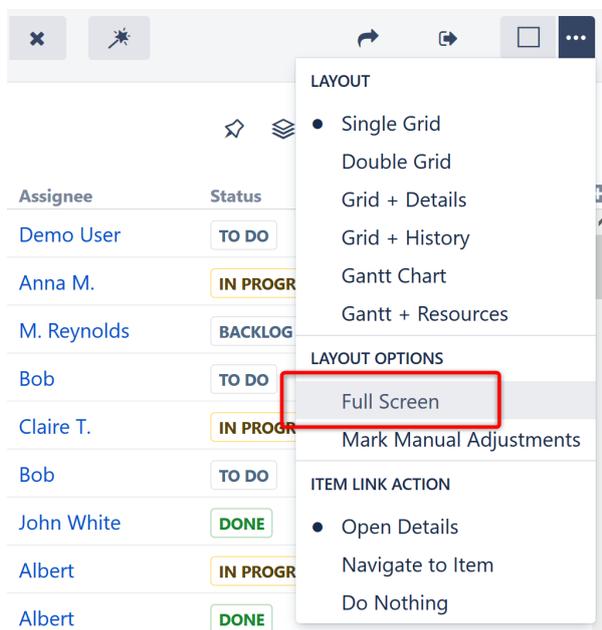
To view the Issue Clipboard, click the structure name or the search-type label at the top of the panel and select **Clipboard** from the menu.



To learn more, see [Using Cut, Copy and Paste](#).

## 2.7.6 Full Screen Mode

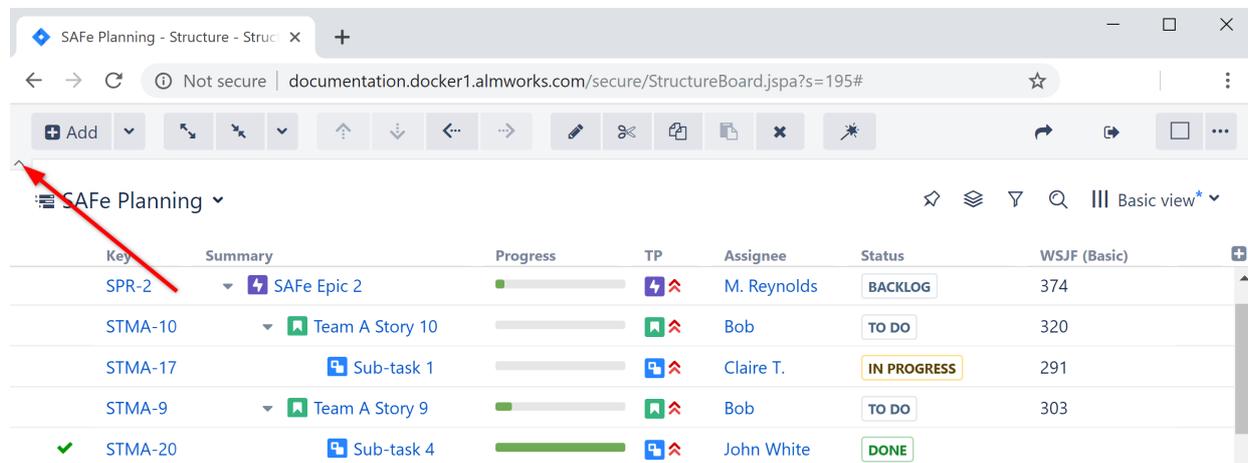
When working with the Structure Board you can turn on Full Screen mode to give more screen space to your data. Full Screen mode can be toggled using the **Toggle Panels** menu or by pressing Z on your keyboard.



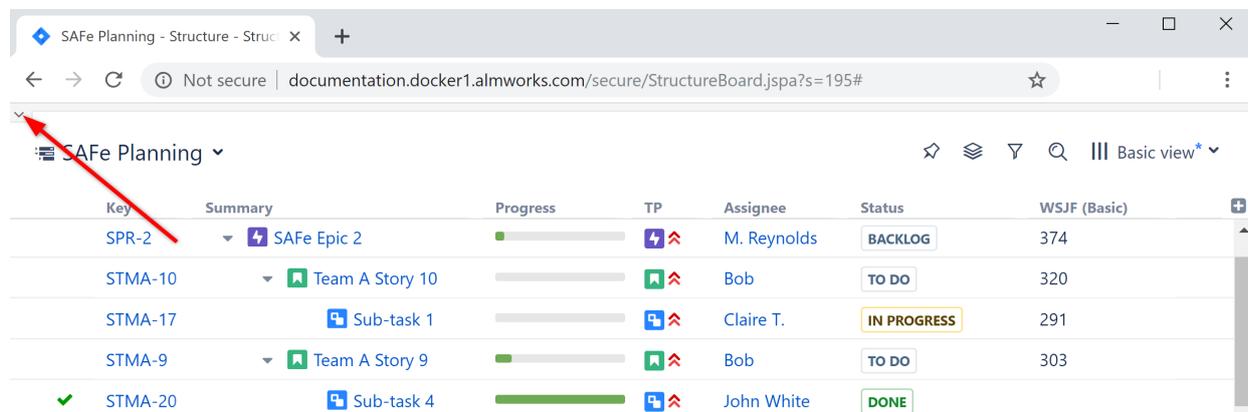
In Full Screen mode, the Jira application header is hidden and the Structure toolbar becomes more compact. To exit Full Screen mode, uncheck **Full Screen** in the Toggle Panels menu or press Z again.

## 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).



## 2.8 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**.

The screenshot shows the Jira Software interface with the 'Structure' dropdown menu open. The 'Manage Structures' option is highlighted with a red box. A red arrow points from the 'Structure' dropdown to the 'Manage Structures' option.

You can also reach the Manage Structures page from the structure selector menu:

The screenshot shows the Jira Software structure selector menu with the 'Manage Structure' option highlighted. A red arrow points from the 'Manage Structure' option to the 'Manage Structures' page.

Structure Name	Σ Story Points	Assignee	Progress	TP
Top-Down Automation	28	Unassigned	Progress bar	⚡ ⬆️
Team B Story 5	5	Unassigned	Progress bar	🛡️ ⬆️
Team B Story 11	11	Nah Duo	Progress bar	🛡️ ⬆️
Team A Story 2	12	Jack Brown	Progress bar	🛡️ ⬆️
minute TV advertisement for prime-time broadc		Man in Black	Progress bar	🛡️ ⬆️
Team C 11		M. Reynolds	Progress bar	⚡ ⬆️
Team C 10	25	Unassigned	Progress bar	⚡ ⬆️
Team A Story 6	25	M. Reynolds	Progress bar	🛡️ ⬆️
Team C 9	54	Bob	Progress bar	⚡ ⬆️
Team B Story 7	5	Nah Duo	Progress bar	🛡️ ⬆️
Team B Story 4	8	Man in Black	Progress bar	🛡️ ⬆️
Team B Story 6	21	Nah Duo	Progress bar	🛡️ ⬆️
STMB-14	17	Unassigned	Progress bar	🛡️ ⬆️

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**
- **My** – lists structures created by you
- **Popular** – lists structures that are marked as favorite by at least 2 users, ordered by their **popularity**
- **Search** – allows you to **find a structure by name, owner or ID**

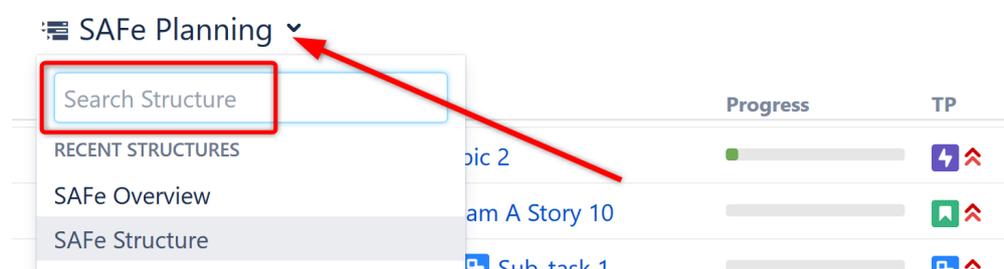
- **All** – lists all structures visible to you
- **Paused** – only displayed when there are structures with [Paused Automation](#)

✓ 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.

More about managing structures:

### 2.8.1 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](#). From the Jira menu, go to **Structure | Manage Structures** and select the **Search** tab.

Manage Structures

- Recent
- Favorite
- My
- Popular
- Search**
- All
- Archived
- Paused

#### Search Structures ?

This page lets you find structures.

[Search by structure parameters](#) Search by the structure ID

---

Name

Searches in the structure's name.

Owner

Start typing to get a list of possible matches.

Permission level

Searches for structures to which you have the specified access level.

Show Archived

## Finding Structures by Name, Access Level or Owner

To search for structures by their properties:

1. Enter any of the search parameters. Parameters are:

<b>Name</b>	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.
<b>Owner</b>	Only structures that are owned by the specified user will be shown. <i>Note: This only works if you have permission to browse users.</i>
<b>Permission Level</b>	Lets you select the structures that you can Edit or Control, according to the selected permission level. (For example, if you select <b>Edit</b> 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.

## 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.

### 2.8.2 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](#) and [Project Pages](#) when the **Auto-switch to default structure** option is selected.

Default structures can be set by:

- Jira administrators can [change the system default structure](#)
- Project managers can select a [project-level default structure](#) for any project that is enabled for Structure

## 2.8.3 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.

The screenshot shows the 'SAFe Planning' dropdown menu with 'FAVORITE STRUCTURES' highlighted. The main table displays the following data:

Structure Name	Progress	TP	Assignee	Status	WSJF (Basic)
Park is Safe' campaign	[Progress Bar]	[TP Icon]	Demo User	TO DO	597
am A Story 1	[Progress Bar]	[TP Icon]	Anna M.	IN PROGRESS	317
opic 2	[Progress Bar]	[TP Icon]	M. Reynolds	BACKLOG	374
am A Story 10	[Progress Bar]	[TP Icon]	Bob	TO DO	320
Sub-task 1	[Progress Bar]	[TP Icon]	Claire T.	IN PROGRESS	291
am A Story 9	[Progress Bar]	[TP Icon]	Bob	TO DO	303
Sub-task 4	[Progress Bar]	[TP Icon]	John White	DONE	
am A Story 8	[Progress Bar]	[TP Icon]	Albert	IN PROGRESS	298
Sub-task 2	[Progress Bar]	[TP Icon]	Albert	DONE	
opic 1	[Progress Bar]	[TP Icon]	Unassigned	SELECTED FOR	400

## Add/Remove Favorite Structures

To favorite a structure, open [Manage Structures](#) and [locate the structure](#) 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.

The 'Manage Structures' page shows the following table of 'Recent Structures':

Name	Owner	Access	Popularity	Operations
★ SAFe Planning Use Transformations to group stories by Sprints, Assignees, Versions, etc to see the work load distribution and do the load balancing	admin (M. Reynolds)	Control	0	Configure Views Delete Archive Import Export Copy
☆ SAFe Overview This structure shows how the plugin can be used as a part of the SAFe implementation.	admin (M. Reynolds)	Control	1	Configure Views Delete

To remove a structure from your favorites list, simply click the star again. To view all favorited structures, open [Manage Structures](#) and click the Favorite tab.

## 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](#) and click the **Popular** tab.

### 2.8.4 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 meaning of the structure to users.
- **Owner** - The owner of the structure. Only Jira administrators can change the owner.
- **Permissions** - Defines who can view, edit or configure the structure. See [Structure Permissions](#) 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](#) for details.
  - *Allow manual adjustments of generated content* - Enables or disables [manual adjustments](#).
- **Time Limit** - Determines the maximum amount of time [Automations](#) can run before being [paused](#).
- **Favorite** - When selected (the star is filled in), the structure will appear in your [Favorite Structures](#) list.

## Edit Structure

### Edit Structure ?

Name\*

Description

Owner    
Start typing to get a list of possible matches.

Permissions User permission level is calculated by applying rules from this list, from top to bottom. **The last matching rule takes precedence.** Structure owner and Jira administrators always have **Control** permissions.

1. By default, permission level is **None**

Add Rule  to  for  [+ Add](#)

Options  Require **Edit Issue** permission on parent issue to rearrange sub-issues  
 Allow manual adjustments of generated content

Time limit  seconds (for automation)

Favorite

[Back](#)

You can specify structure details when [Creating New Structures](#) and when [Editing Structure Details](#).

### Editing Structure Details

The Edit Structure screen allows you edit [details](#) 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.

## Manage Structures

Recent	Recent Structures <span style="float: right;">?</span>				
Favorite	This page lets you manage your recently used structures.				
My					
Popular	Name	Owner	Access	Popularity	Operations
Search	<a href="#">☆ SAFe Planning</a> Use Transformations to group stories by Sprints, Assignees, Versions, etc to see the work load distribution and do the load balancing	admin (M. Reynolds)	Control	0	<a href="#">Configure</a> <a href="#">Views</a> <a href="#">Delete</a> <a href="#">Archive</a> <a href="#">Import</a> <a href="#">Export</a> <a href="#">Copy</a>
All	<a href="#">☆ SAFe Overview</a> This structure shows how the plugin can be used as a part of the SAFe implementation.	admin (M. Reynolds)	Control	1	<a href="#">Configure</a> <a href="#">Views</a> <a href="#">Delete</a> ...
Archived					
Paused					



If you do not see a **Configure** link, you probably do not have Control permission for that structure.

See [Structure Details](#) for more information.

### 2.8.5 Customizing View Settings

A structure's view settings determine which views are offered to the users in the [Views Menu](#) and which view is used by default. To customize view settings for a structure, open the [Manage Structures](#) page, locate the structure and click the **Views** link.

Views for Structure "SAFe Planning" ?

A view defines columns displayed in the Structure grid. View settings allow you to choose which views are offered in the **Views** menu for this structure and which views are used by default (if the user hasn't chosen another view). The menu and the defaults can be customized for different Jira pages. If views are not customized for the structure, global default settings apply.

Structure **SAFe Planning** (ID: 195)

Use Transformations to group stories by Sprints, Assignees, Versions, etc to see the work load distribution and do the load balancing

View Settings  Default  
 Customized

**Views Menu**

⌵	<b>Basic view</b> Offered on: All pages ▾	✕ Remove
⌵	<b>Planning</b> Offered on: Structure Board, Structure with Issue Details, Issue Page, Project Pages ▾	✕ Remove
⌵	<b>Tracking</b> Offered on: All pages ▾	✕ Remove
⌵	<b>Compact</b> Offered on: Structure with Issue Details ▾	✕ Remove

Add view:

**Default Views**

Structure Board:	<input type="text" value="Basic view"/>
Structure with Issue Details:	<input type="text" value="Compact"/>
Issue Page:	<input type="text" value="Basic view"/>
Project Pages:	<input type="text" value="Basic view"/>



You must have **Control** permissions for the structure to adjust its view settings.

## 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.

## Configuring Views Menu

The **Views Menu** section lets you select which views appear under the Associated Views section of the [Views Menu](#) for each type of [Jira page](#).

- 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.

## 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**.

### 2.8.6 Structure Permissions

Every structure has a list of permission rules, which define who is allowed to see, edit or configure the structure.

#### Access Levels

Each user has one of the following access levels to a structure:

<b>None</b>	The user does not see the structure at all and does not know that it exists.
-------------	--

<b>View</b>	The user can view the structure but cannot make changes.
<b>Edit</b>	The user can view the structure and can rearrange, add and remove issues from the structure. The user cannot, however, create or modify <a href="#">generators</a> .
<b>Automate</b>	The user has full edit access to the structure, including modifying <a href="#">generators</a> and <a href="#">effectors</a> .
<b>Control</b>	The user can view, edit and configure the structure - including changing structure permission rules.

## 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.

## Permission Rules

Users who have **Control** permission for a structure can define permission rules by [Editing Structure Details](#).

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).

**Permissions** User permission level is calculated by applying rules from this list, from top to bottom. **The last matching rule takes precedence.** Structure owner and Jira administrators always have **Control** permissions.

- ↓ 🗑️ 1. **View** for **Everyone**
- ↑ ↓ 🗑️ 2. **Edit** for **jira-users** (Group)
- ↑ ↓ 🗑️ 3. **Automate** for **jira-developers** (Group)
- ↑ 🗑️ 4. **Control** for **jira-administrators** (Group)

Add Rule Set Permission Level ▼ to Control ▼ for Group ▼ jira-administrators ▼ + Add

### Permission settings

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:

<b>Everyone</b>	Matches any user, including anonymous (not logged in). This condition can be used to set a default permission for everyone.
<b>Group(G)</b>	Matches users that belong to the group G.
<b>Project Role(R,P)</b>	Matches users that have role R in project P.
<b>User</b>	Adds the permission for a specific Jira user.



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](#) 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:

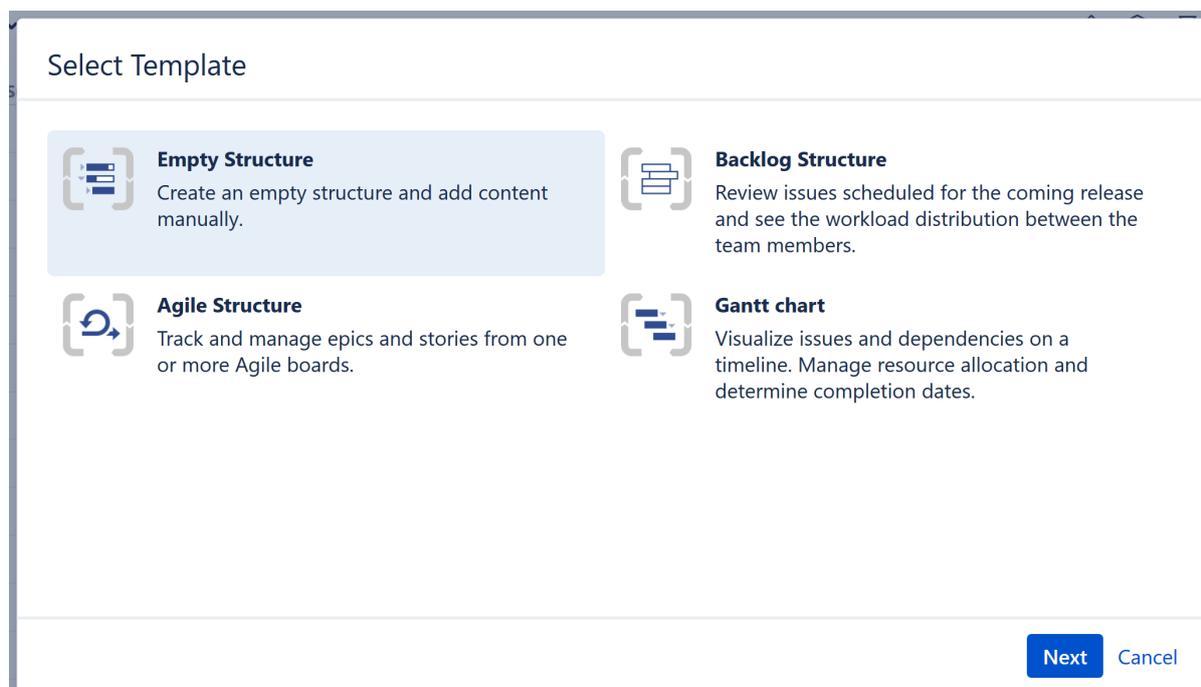
- Top-level issues do not have parent issues, and therefore are not affected by this flag: the user can add/rearrange issues at the top level of the structure if they have Edit access level.
- The Edit Issue permission applies only to the direct parent issue. If issue A has sub-issue B, and B has sub-issue 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](#)).

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.

## 2.8.7 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](#) 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 name of the new structure. You can optionally add a description and select users to share the structure with. These users will have Edit permissions, but you can change this setting or add additional permissions later using the [Structure Details](#) page.

**i** 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](#).

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](#) and [organize](#) issues or apply [automations](#) once the structure is created.

**i** Only logged-in users who have access to Structure are allowed to create new structures. See [Who Has Access to the Structure](#) for more details.

## 2.8.8 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](#) to copy a part of the structure.

### Create a Copy

To create a copy of a structure, open the [Manage Structures](#) page using the top navigation **Structure** menu. Find the structure you'd like to copy and click the **Copy** link in the Operations column.

Manage Structures

**Recent**

- Favorite
- My
- Popular
- Search
- All
- Archived
- Paused

Recent Structures ?

This page lets you manage your recently used structures.

Name	Owner	Access	Popularity	Operations
★ <a href="#">SAFe Planning</a> Use Transformations to group stories by Sprints, Assignees, Versions, etc to see the work load distribution and do the load balancing	admin (M. Reynolds)	Control	1	<a href="#">Configure</a> <a href="#">Views</a> <a href="#">Delete</a> <a href="#">Archive</a> <a href="#">Import</a> <a href="#">Export</a> <a href="#">Copy</a>



If you don't see **Copy** in the Operations column, you probably [do not have permissions](#) 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](#), and [synchronizers](#) it contains. If the structure contains [automation](#), you can click the **Calculate** link in the **Visible Content** section to execute the generators and see the generated content statistics.

## Copy Structure "SAFe Planning"

Structure	SAFe Planning (ID 195)
Description	Use Transformations to group stories by Sprints, Assignees, Versions, etc to see the work load distribution and do the load balancing
Owner	admin (M. Reynolds)
Non-automated Content	All items: 4 Issues: 0 Generators: 4 (1 inserter, 1 extender, 1 grouper, 1 filter) Does not include generated parts.
Visible Content	All items: Not calculated Issues: Not calculated <a href="#">Calculate</a>
Generators	<input type="text" value="Copy generators to the resulting structure"/>
Clone Issues?	<input checked="" type="radio"/> No — the new structure will contain the same issues as the original structure <input type="radio"/> Yes — the new structure will contain copies (clones) of the issues from the original structure
	<input type="button" value="Copy Structure"/> <input type="button" value="Cancel"/>

Select the following options for copying the structure:

- 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.
- Choose if you'd like to copy [synchronizers](#)**, 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](#) to create synchronizers. See [Copying Synchronizers](#) for more details.
- Choose if you'd like to clone issues** (see [Copy vs. Clone](#)). If yes, [enter the additional parameters for the cloning process](#).

Once you've made you selections, press **Copy Structure** or **Start Cloning**.

## 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](#).

You can immediately edit the new structure's properties on the screen with the copy result.

## 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 <b>Clone Issues?</b>	<b>No</b>	<b>Yes</b>
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:	<b>View</b> access to the original structure <b>Create Structure</b> permission	<b>View</b> access to the original structure <b>Create Structure</b> permission <b>Bulk Change</b> global Jira permission A number of project-level permissions

For details about configuring and running cloning, see [Copying Structure and Cloning Issues](#).

## Copying Structure and Cloning Issues

When [copying a structure](#), 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.



On the rare occasion when permissions or other restrictions are changed while the cloning operation is in progress, the operation may still fail after the initial checks.

## Cloning Parameters

When Clone Issues is selected, you can specify a number of additional parameters.

## Issue Cloning Parameters

Structure plugin will create a copy of each issue in the structure. Each new issue will have the initial status (according to its workflow) and all applicable fields copied from the original issue.

Create in Project

By default, each clone is created in the same project as the original issue. If Target Project is selected, issue clones will be created in the specified project.

Summary Prefix

If set, this text will be prepended to the summary of each clone.

Summary Suffix

If set, this text will be appended to the summary of each clone.

Labels

Optionally, enter labels (separated by space) to be added to each clone.

Link Back  Link each clone to its original issue  
Link type: cloned issue  original issue

Options  Copy comments  
 Copy attachments  
 Clone Jira sub-tasks of the cloned issues (even if the sub-tasks are not in the structure)  
 Copy issue links  
 Copy watchers

Notifications  Send e-mail notifications for cloned issues

<b>Create in Project</b>	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.
<b>Summary Prefix</b> <b>Summary Suffix</b>	Lets you modify the summary of the clones. If the resulting summary gets longer than the Jira limit (255 characters), it will be truncated.
<b>Labels</b>	Lets you add space-delimited labels to the cloned issues. (Already existing labels are preserved.)
<b>Link Back</b>	If specified, every new issue will be linked with its original issue.
<b>Copy comments</b>	If selected, all comments are also copied. If not selected, new issues will have no comments.
<b>Copy attachments</b>	If selected, attachments are copied (the actual files are copied on the Jira server).

<b>Clone Jira sub-tasks of the cloned issues</b>	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.
<b>Copy issue links</b>	<p>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.</p> <div data-bbox="405 667 1428 835" style="border: 1px solid #f9c77d; padding: 10px;"> <p> If you use the <b>Link Back</b> option, then the links of the type selected for linking back to original issues will <b>not be copied</b>.</p> </div> <p>Scrum board Epic-Story relationships are also copied when you select this option. The rule is the same as for issue links:</p> <ul style="list-style-type: none"> <li>• If you clone an epic together with its stories, the cloned stories will be added to the cloned epic.</li> <li>• If you clone the stories alone, the clones will be added to the original epic.</li> </ul>
<b>Copy watchers</b>	If selected, the users watching an original issue will be added to the watcher list of the clone.
<b>Notifications</b>	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](#).)
- 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](#) page and locate the original structure. It should show that the structure is being copied.

Manage Structures

**Recent**

- Favorite
- My
- Popular
- Search
- All
- Archived
- Paused

Recent Structures ?

This page lets you manage your recently used structures.

Name	Owner	Access	Popularity	Operations
★ <a href="#">Manually Built Structure</a>	admin (M. Reynolds)	Edit generators	1	<a href="#">Copy</a>

This structure is being copied | **76% complete**

Click to open the cloning progress page

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](#) 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.

### 2.8.9 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.

**i** *Read-only* means that users cannot add, remove or move items (issues, folders, memos, generators) in the archived structure.

## Archive a Structure

To archive a structure:

1. Open the [Manage Structures](#) page using the top navigation **Structure** menu.

2. Find the structure you'd like to archive and click the **Archive** link in the Operations column.
3. Review the structure you are about to archive and confirm the operation. You can **Unarchive** the structure in the future.

Manage Structures

**Recent**

- Favorite
- My
- Popular
- Search
- All
- Archived
- Paused

Recent Structures ?

This page lets you manage your recently used structures.

Name	Owner	Access	Popularity	Operations
<p>★ <a href="#">SAFe Planning</a></p> <p>Use Transformations to group stories by Sprints, Assignees, Versions, etc to see the work load distribution and do the load balancing</p>	admin (M. Reynolds)	Control	1	<ul style="list-style-type: none"> <li>Configure</li> <li>Views</li> <li>Delete</li> <li><b>Archive</b></li> <li>Import</li> <li>Export</li> <li>Copy</li> </ul>



You need **Control** access level to be able to archive a structure.



If there are any synchronizers installed for the structure you archive, they will be disabled.

## 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](#) 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, click the **Unarchive** link in the Operations column.

Manage Structures

Recent  
Favorite  
My  
Popular  
Search  
All  
**Archived**  
Paused

Archived Structures ?

This page shows all archived structures available to you.

Name	Owner	Access	Popularity	Operations
★ <b>SAFe Planning</b> Use Transformations to group stories by Sprints, Assignees, Versions, etc to see the work load distribution and do the load balancing	admin (M. Reynolds)	Control	1	<a href="#">Configure</a> <a href="#">Views</a> <a href="#">Delete</a> <a href="#">Unarchive</a> <a href="#">Import</a> <a href="#">Export</a> <a href="#">Copy</a>

i You need **Control** access level to be able to unarchive a structure.

## 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.

☰ Sort Test ▾ **ARCHIVED**

Summary	Assignee	Progress	TP	Story Points	Key
•  Story 3.2	Bob Robertson	<div style="width: 50%;"></div>	=		VIP-10
Story 3.1	Claire Clayton	<div style="width: 50%;"></div>	=		VIP-9
Story 2.2	Bob Robertson	<div style="width: 100%;"></div>	=	16	VIP-8
Story 2.1	Claire Clayton	<div style="width: 100%;"></div>	=	16	VIP-7
Story 1.3	Bob Robertson	<div style="width: 100%;"></div>	=	16	VIP-6

## Synchronizers and Archiving

If there are any synchronizers installed for the structure you archive, they will be disabled. After unarchiving, we recommend that you review the synchronizers configuration and resync prior to enabling them.

Until the structure is unarchived, you cannot resync and enable synchronizers. However, you can [Export](#) an archived structure, if you have a special permission to control synchronizers.

## 2.8.10 Deleting a Structure

 Deleting a structure cannot be undone. If there is a chance you may need the structure in the future, consider [archiving](#) 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!**

Manage Structures

**Recent**

Favorite

My

Popular

Search

All

Archived

Paused

Recent Structures 

This page lets you manage your recently used structures.

Name	Owner	Access	Popularity	Operations
<p>★ <a href="#">SAFe Planning</a></p> <p>Use Transformations to group stories by Sprints, Assignees, Versions, etc to see the work load distribution and do the load balancing</p>	admin (M. Reynolds)	Control	1	<p><a href="#">Configure</a></p> <p><a href="#">Views</a></p> <p><a href="#">Delete</a></p> <p><a href="#">Archive</a></p> <p><a href="#">Import</a></p> <p><a href="#">Export</a></p> <p><a href="#">Copy</a></p>



You need **Control** access level to be able to delete a structure. See [Structure Permissions](#) to learn more.

### 2.8.11 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.

### 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](#).

### Creating Issues and a Structure from Template

Once you have a template structure, you can use [Copy](#) action from the **Manage Structures** page and turn on [Clone Issues](#) option. For details about configuring and running cloning operation, refer to [Copying Structure and Cloning Issues](#) 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](#) on all issues - for example, to remove a template marker.

## Template Projects

In the same manner, you can create a template project with template issues, and put them all into a template structure.

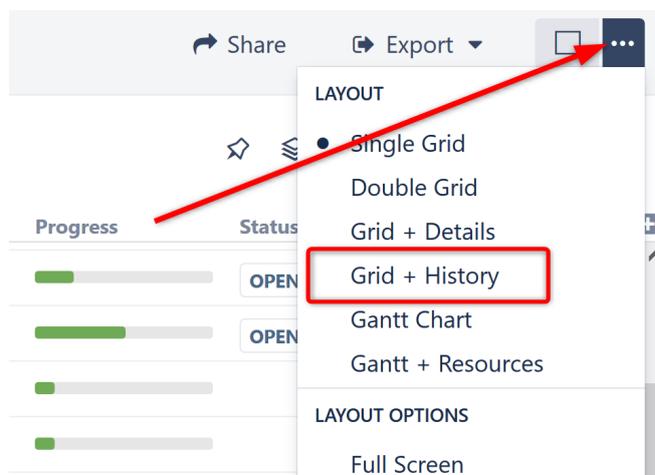
When you need to create a new project based on the template project, do the following:

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.

### 2.8.12 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](#) will not appear in the history. However, the addition, moving and removal of the generators themselves are recorded.

### 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.

Key	Summary	Progress	TP	Status	WSJF (Basic)	Index
STMB-40	Epic 1	<div style="width: 100%;"></div>		TO DO	800	1
STMB-31	Story 1	<div style="width: 100%;"></div>		IN PROGRESS	326	1.1
STMB-32	Story 2	<div style="width: 100%;"></div>		TO DO	296	1.2
STMB-43	Bug 1	<div style="width: 100%;"></div>		DONE		1.2.1
STMB-42	Story 3	<div style="width: 100%;"></div>		IN PROGRESS	303	1.3
STMB-41	Epic 2	<div style="width: 100%;"></div>		TO DO	144	2
STMB-37	Story 4	<div style="width: 100%;"></div>		IN PROGRESS	326	2.1
STMB-39	Story 6	<div style="width: 100%;"></div>		TO DO	338	2.2

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.

## 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 [Automation](#) - 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.)

## Printing a Previous Structure Version

You can [create a printable version](#) 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.).

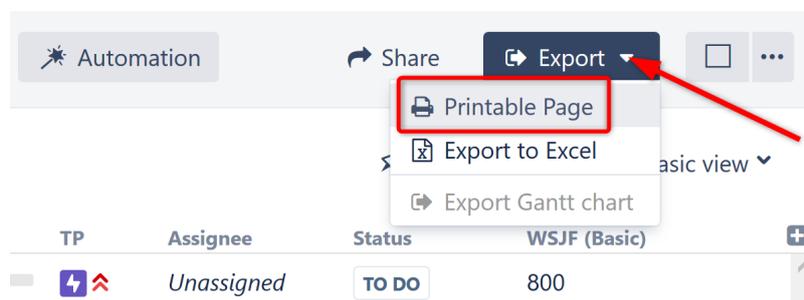
## Exporting a Previous Structure Version to Excel

You can also [export the structure to Excel](#). 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.).

### 2.8.13 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. Note: if [horizontal scrolling](#) is turned on, columns will be exported in their original, non-horizontal scrolling order.
- 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.

Key	Summary	Σ Story Points	Σ Time Spent	Progress	TP	Assignee	Status	WSJF (Basic)
STMB-40	Epic 1		3d 3h	31% <div style="width: 31%;"></div>		Unassigned	TO DO	800
STMB-31	Story 1		2d 4h	45% <div style="width: 45%;"></div>		Unassigned	IN PROGRESS	326
STMB-32	Story 2		7h	16% <div style="width: 16%;"></div>		Unassigned	TO DO	296
STMB-43	Bug 1		2h	100% <div style="width: 100%;"></div>		Unassigned	DONE	

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.

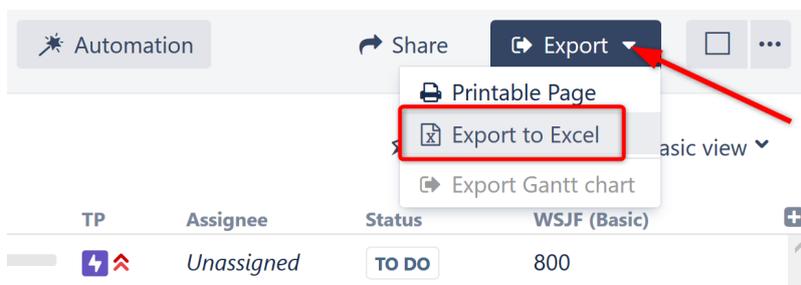


It's a good idea to print a single sample page to make sure font size is big/small enough.

### 2.8.14 Exporting Structure to XLS (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**

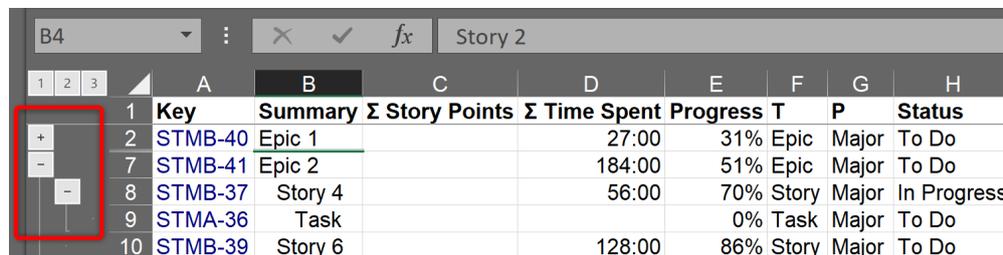


The browser will download a new XLS file, which you can save or open. The XLS file will:

- Contain all the issues that are present in the structure.
- Preserve the structure's hierarchy.
- Include the same columns as the structure.
- Include columns in non-horizontal scrolling order, even if [horizontal scrolling](#) is turned on.
- 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.



1	2	3	A	B	C	D	E	F	G	H	
			1	<b>Key</b>	<b>Summary</b>	<b>Σ Story Points</b>	<b>Σ Time Spent</b>	<b>Progress</b>	<b>T</b>	<b>P</b>	<b>Status</b>
			2	STMB-40	Epic 1		27:00	31%	Epic	Major	To Do
			7	STMB-41	Epic 2		184:00	51%	Epic	Major	To Do
			8	STMB-37	Story 4		56:00	70%	Story	Major	In Progress
			9	STMA-36	Task			0%	Task	Major	To Do
			10	STMB-39	Story 6		128:00	86%	Story	Major	To Do

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](#).

## 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.

Column Type	Notes
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 <b>Format Cells   Alignment   Wrap Text</b> 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 Worked	Duration fields contain actual numbers (fractional number of days), which you can sum or otherwise process. The display format is <b>HH:MM</b> , where HH is the number of hours and MM is the number of minutes. So an estimation of 5 days will be displayed as <b>40:00</b> (if you have 8-hour work days).
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

### 2.8.15 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 [Automation](#), 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.

### 2.8.16 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.

## 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**.

## 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.

The screenshot shows the 'Activity Stream' interface. It lists three activities:

- Item 1:** Demo Account moved MARS-4 in Mars Colony upwards under MARS-1 (4 minutes ago).
- Item 2:** Demo Account added MARS-3005 to Mars Colony under MARS-2464 (2 minutes ago).
- Item 3:** Igor Sereda added MARS-3006 to Mars Colony under MARS-2464 (Moments ago).

A red arrow points from the text 'Click to open Structure History' to the 'Moments ago' timestamp of the third item.

*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](#).

## 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.

## 2.9 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 Mac OS X and PC keyboards.

### 2.9.1 Keyboard Shortcuts (PC)

#### Navigation

Action	Shortcut
Select Issue	<i>Left-Click</i>
Show/Hide Issue Details	o
Previous Issue	ork
Next Issue	orj
Expand Sub-Issues	
Collapse Sub-Issues	
For Large Structure	PgUp PgDn Home End
Add Column	tt
Expand All	++
Collapse All	--

#### Changing Structure

Action	Shortcut
Move Up	Ctrl+
Move Down	Ctrl+
Indent	Ctrl+
Outdent	Ctrl+
Drag and Drop	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+

## Structure Views

Action	Shortcut
Switch View	vv
Save View	vs
Save View As	vss
Revert Changes to View	vr

## 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
Assign Issue	a
Assign to Me	i
Edit Issue Labels	l
Actions Drop-Down	Alt+

## Changing Issues

Action	Shortcut
Edit Field	<i>Double-Click</i>
Edit Summary	Tab <i>or</i> F2
Finish & Save	Enter <i>or</i> Ctrl+Enter
Cancel Field Changes	Esc
Edit Next Field	Tab <i>or</i> Ctrl+Alt+
Edit Previous Field	Shift+Tab <i>or</i> 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	\
<a href="#">View Full-Size Image</a>	ii
Show/Hide Issue Details without Switching Panel	Shift+o
Show Automation	~

## 2.9.2 Keyboard Shortcuts (Mac)

### Navigation

Action	Shortcut
Select Issue	<i>Left-Click</i>
Show/Hide Issue Details	o
Previous Issue	<i>or</i> k
Next Issue	<i>or</i> j
Expand Sub-Issues	
Collapse Sub-Issues	
For Large Structure	
Add Column	t t
Expand All	++
Collapse All	-
Change Structure	xx

### Structure Views

Action	Shortcut
Switch View	v v
Save View	v s
Save View As	v s s

### Changing Structure

Action	Shortcut
Move Up	
Move Down	
Indent	
Outdent	
Drag and Drop	Drag
New Issue	
New Sub-Issue	
Remove from Structure	

### Changing Issues

Action	Shortcut
Edit Field	<i>Double-Click</i>
Edit Summary	tab
Finish & Save	<i>or</i>
Cancel Field Changes	esc
Edit Next Field	tab <i>or</i>
Edit Previous Field	tab <i>or</i>

Action	Shortcut
Revert Changes to View	vr

## 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	

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

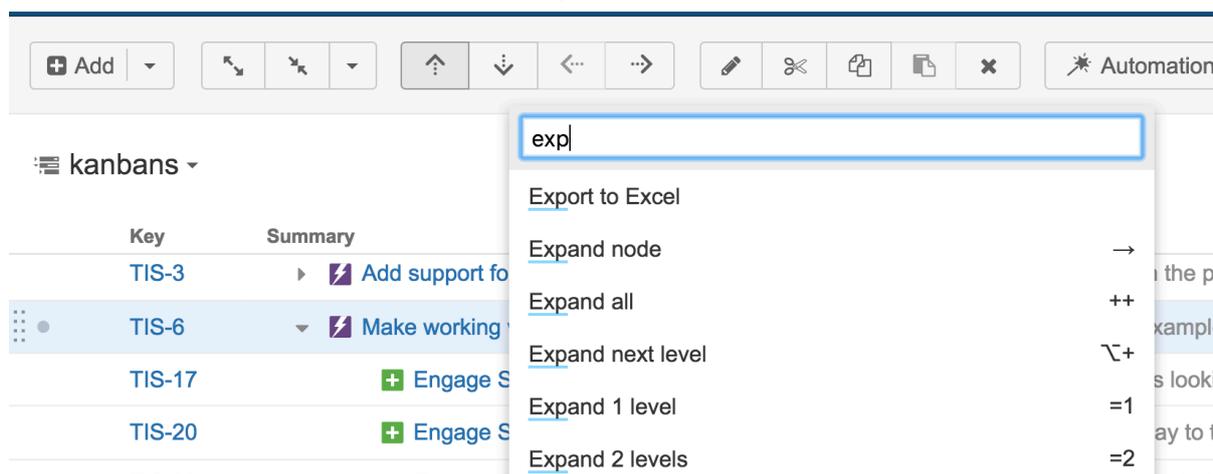
Action	Shortcut
Hide/Show Resolved	rr
Cut (Prepare to Move)	x
Paste (Move)	v
Paste Sub-Issue (Move)	v
Fix/Unfix View on Issue	.

Action	Shortcut
View Full-Size Image	ii
Show/Hide Issue Details without Switching Panel	o

### 2.9.3 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.

## 2.10 Structure Gadget

Structure gadgets can be added to the Jira dashboard or Confluence pages.

See the following articles to learn more.

### 2.10.1 Dashboard Gadget

You can view and edit structures directly from your Jira dashboard with the Structure dashboard gadget.

## 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.

## 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.)

**Structure**

Structure\*

View\*  New View...

Filter Type

Title

Visible Rows   
Maximum number of visible rows (up to 50).

Allow changes (subject to permissions)

Alternative settings when maximized

	Key	Summary	Progress	Assignee	
⋮ ●8	TP-124	▶ 🧑 Site preparations	<div style="width: 80%; height: 10px; background-color: #76923c;"></div>	Bob	⚙️
	TP-31	▶ 🧑 Marketing + PR activities	<div style="width: 60%; height: 10px; background-color: #76923c;"></div>	Bob	
	TP-8	▶ 🧑 Rides + attractions	<div style="width: 40%; height: 10px; background-color: #76923c;"></div>	Harry	

Showing 58 items 🔗 Open

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](#) 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](#). You can choose between a text filter, JQL, S-JQL, or a saved JQL filter – see [Search](#).
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**.

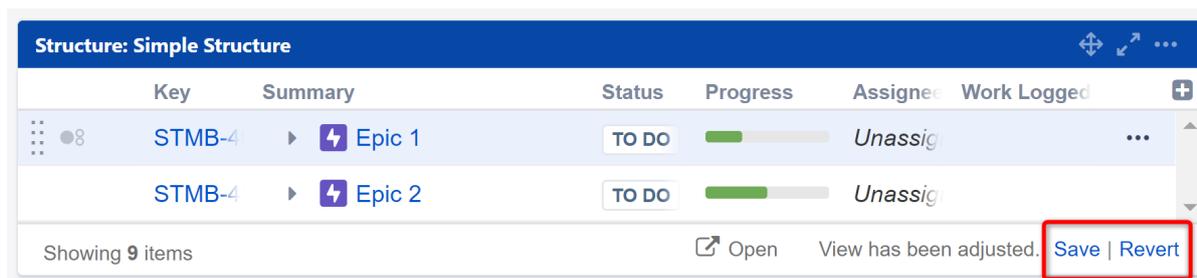


Deselect **Allow Changes** to protect the structure from accidental changes, such as changes caused by drag-and-drop or hitting the Delete key.

## Customizing Gadget View

Once you have created your gadget, you can customize by adding, removing or rearranging columns – see [Customizing Columns](#).

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.

**i** You must have [Update permission](#) for the current view to save changes.

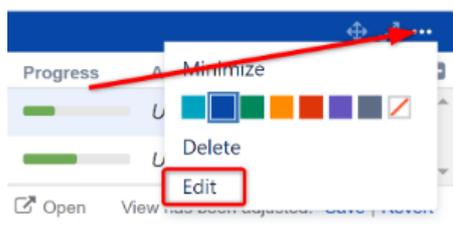
## Revert

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](#) 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.

## 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](#)

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

## Open on Structure Board

Working from the [Structure Board](#) provides the most unrestricted Structure experience. To get to the Structure Board from a gadget, click the **Open** link.

**Structure**

Manually Built Structure ▾

Key	Summary	Progress	TP	Assignee
	Theme Park Construction	<div style="width: 50%;"></div>		
TP-124	Site preparations	<div style="width: 75%;"></div>		Bob
SP-9	Build a transparent dome	<div style="width: 75%;"></div>		Bob
QA-7	Check seismic activity	<div style="width: 25%;"></div>		Unassigned

Showing 4 items

[Open](#) Info

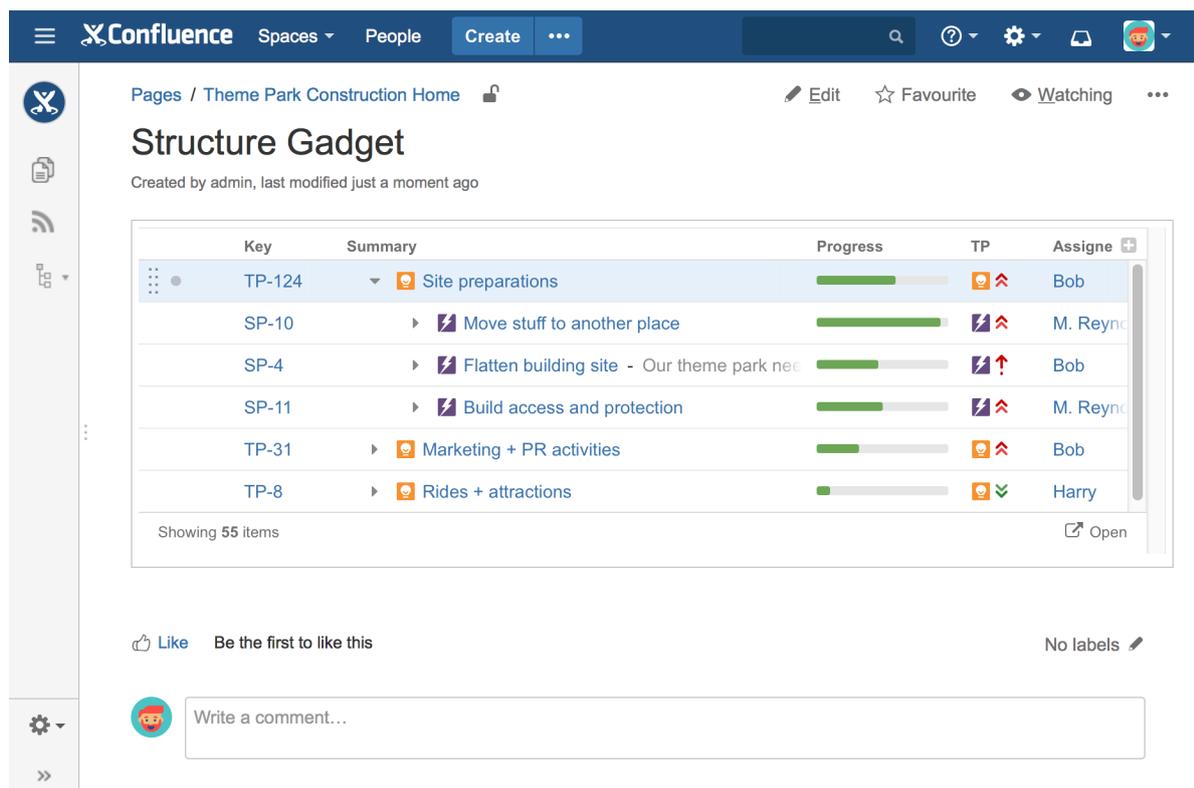
The structure will open on the Structure Board with the same [filters](#) and [transformations](#) that were applied in the original gadget. For example, if the gadget only shows items from a specific project, sorted by Assignee, that's exactly what you'll see on the Structure Board. To review or remove these transformations, click the Transformations button



in the panel toolbar.

### 2.10.2 Confluence Gadget

You can embed a Structure gadget into a Confluence page to view or edit the structure in Confluence.



Pages / Theme Park Construction Home

## Structure Gadget

Created by admin, last modified just a moment ago

Key	Summary	Progress	TP	Assignee
TP-124	Site preparations	<div style="width: 100%;"></div>		Bob
SP-10	Move stuff to another place	<div style="width: 100%;"></div>		M. Reyno
SP-4	Flatten building site - Our theme park nee	<div style="width: 100%;"></div>		Bob
SP-11	Build access and protection	<div style="width: 100%;"></div>		M. Reyno
TP-31	Marketing + PR activities	<div style="width: 100%;"></div>		Bob
TP-8	Rides + attractions	<div style="width: 100%;"></div>		Harry

Showing 55 items

Like Be the first to like this No labels

Write a comment...



Before you can use a Structure gadget on a Confluence page, your Confluence administrator must [add Structure gadget to the Confluence configuration](#). If you try to insert a macro and don't see *Structure* in the available list, most likely the gadget is not configured.

## 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.

### Insert 'Structure' Macro

Displays issue structure and lets you edit the hierarchy and the issues.

Use the parameters below to configure your gadget's properties.

This gadget may also have configurable properties that can only be accessed on the gadget itself. Use the gadget preview on the left to access them.

Width

Examples: "300" (300px), "300px", "50%", "auto"

**Border**  
Whether or not to surround the gadget with a thin border

↻ **Preview**

Structure\*

View\*

Filter **No Filter selected**

Title

Visible Rows   
Maximum number of visible rows (up to 50).

**Allow changes (subject to permissions)**

Alternative settings when maximized

Select macro
⚠ Please complete the configuration in the preview area first

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** 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](#). You can choose between a text filter, JQL, S-JQL, or a saved JQL filter – see [Search](#).
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 Confluence users 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**.

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.

## 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 the Structure gadget – see [Editing from Gadget](#)
- The displayed Structure gadget is not suitable for printing. Support for a printable Structure gadget is coming later. For now, please use [Printable Page](#) to print a structure separately.
- In the current version the Structure Gadget in Confluence is supported, but some issue may occur. This will be fixed in the future versions.

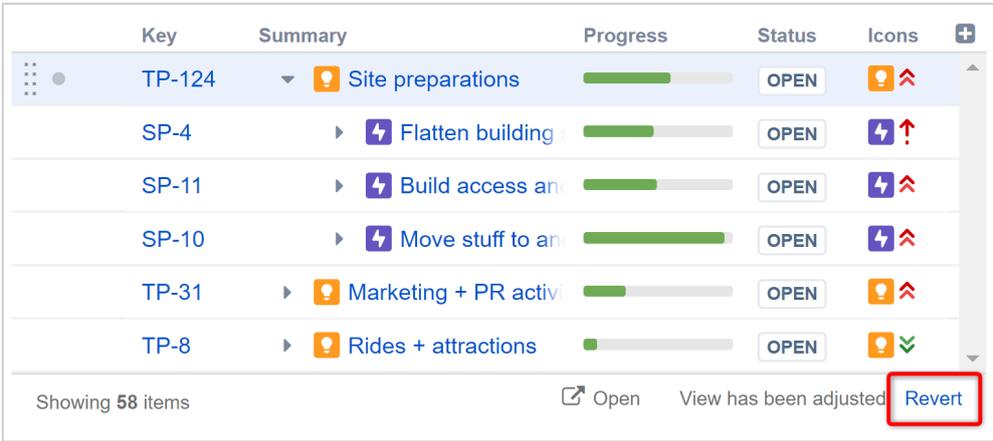
However, many features are still available, including

- 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

## Customizing Gadget View

Once you have created your gadget, you can customize by adding, removing or rearranging columns – see [Customizing Columns](#).

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.



Key	Summary	Progress	Status	Icons
TP-124	Site preparations	<div style="width: 100%;"><div style="width: 100%;"></div></div>	OPEN	🔧 ⬆️
SP-4	Flatten building	<div style="width: 100%;"><div style="width: 100%;"></div></div>	OPEN	⚡ ⬆️
SP-11	Build access and	<div style="width: 100%;"><div style="width: 100%;"></div></div>	OPEN	⚡ ⬆️
SP-10	Move stuff to an	<div style="width: 100%;"><div style="width: 100%;"></div></div>	OPEN	⚡ ⬆️
TP-31	Marketing + PR activ	<div style="width: 100%;"><div style="width: 100%;"></div></div>	OPEN	🔧 ⬆️
TP-8	Rides + attractions	<div style="width: 100%;"><div style="width: 100%;"></div></div>	OPEN	🔧 ⬆️

Showing 58 items Open View has been adjusted Revert

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.

### Edit 'Structure' Macro

Displays issue structure and lets you edit the hierarchy and the issues.

Use the parameters below to configure your gadget's properties.

This gadget may also have configurable properties that can only be accessed on the gadget itself. Use the gadget preview on the left to access them.

Width  
450

Examples: "300" (300px), "300px", "50%", "auto"

Border  
Whether or not to surround the gadget with a thin border

Preview

Key	Summary	Progress	Status	Icon	
TP	Site preparations	<div style="width: 100%;"></div>	OP		
SP	Flatten building	<div style="width: 100%;"></div>	OP		
SP	Build access an	<div style="width: 100%;"></div>	OP		
SP	Move stuff to an	<div style="width: 100%;"></div>	OP		
TP	Marketing + PR activ	<div style="width: 100%;"></div>	OP		
TP	Rides + attractions	<div style="width: 100%;"></div>	OP		

58 items

Open Save View Revert

Select macro

Save Cancel

**i** You must have [Update permission](#) 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](#) for details.
2. From the Edit 'Structure' Macro screen, open the gadget configuration by clicking the edit icon.

Key	Summary	Progress	Status	Icon	
TP	Site preparations	<div style="width: 100%;"></div>	OP		
SP	Flatten building	<div style="width: 100%;"></div>	OP		
SP	Build access an	<div style="width: 100%;"></div>	OP		
SP	Move stuff to an	<div style="width: 100%;"></div>	OP		
TP	Marketing + PR activ	<div style="width: 100%;"></div>	OP		
TP	Rides + attractions	<div style="width: 100%;"></div>	OP		

58 items

Open Save View | Revert

3. Click the **New View** button, located beside the view selector. An additional form appears – enter the new view name and click **Create View**.

Structure\*

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**.

View\*

**i** This view is not public. If a user looking at this gadget does not have access to this view, a default view is used instead.



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.

## Open on Structure Board

Working from the [Structure Board](#) provides the most unrestricted Structure experience. To get to the Structure Board from a gadget, click the **Open** link.

## Structure Gadget

Created by Demo User, last modified just a moment ago

Key	Summary	Progress	Status	Assign	Icons	+
TP-124	Site preparations	<div style="width: 80%;"></div>	OPEN	Bob	🔔 ⬆️	
SP-4	Flatten building	<div style="width: 80%;"></div>	OPEN	Bob	🔔 ⬆️	
SP-11	Build access an	<div style="width: 80%;"></div>	OPEN	M. Re	🔔 ⬆️	
SP-10	Move stuff to an	<div style="width: 80%;"></div>	OPEN	M. Re	🔔 ⬆️	
TP-31	Marketing + PR activ	<div style="width: 80%;"></div>	OPEN	Bob	🔔 ⬆️	
TP-8	Rides + attractions	<div style="width: 80%;"></div>	OPEN	Harry	🔔 ⬆️	

Showing 58 items

🔗 Open

The structure will open on Structure Board with the same [filters](#) and [transformations](#) 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



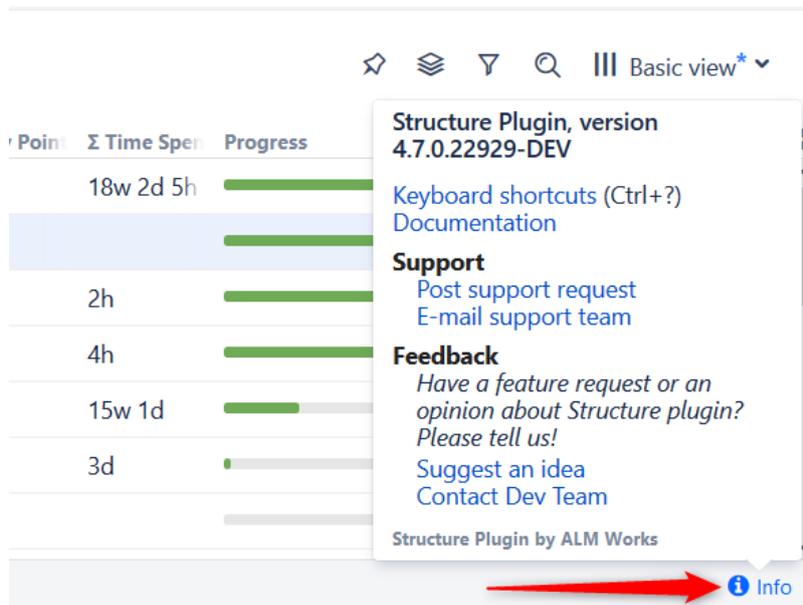
in the panel toolbar.

## 2.11 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](#) or contact our support team.

### 2.11.1 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 a list of keyboard shortcuts, our resource documentation, and our support services.



### 2.11.2 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](#)
- [Suggest an idea](#) on our UserVoice forum
- [Write to us](#) just to say hi or with any comments or questions

## 3 Structure 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](#)
2. [Setting Up Structure License](#)
3. [Getting Started with Structure](#)

Contents:

### 3.1 Installing Structure

Structure is installed like most other apps.

1. Before installing Structure in production, make sure your Jira meets the [Memory Guidelines](#).
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](#) 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](#).

Congratulations! You can now spread the word and help users get started with Structure – see [Getting Started with Structure](#).

ⓘ **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](#) for possible causes and solutions.

### 3.1.1 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.

### 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](#) 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.

### 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.

## 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, [Automation](#) will not work there.

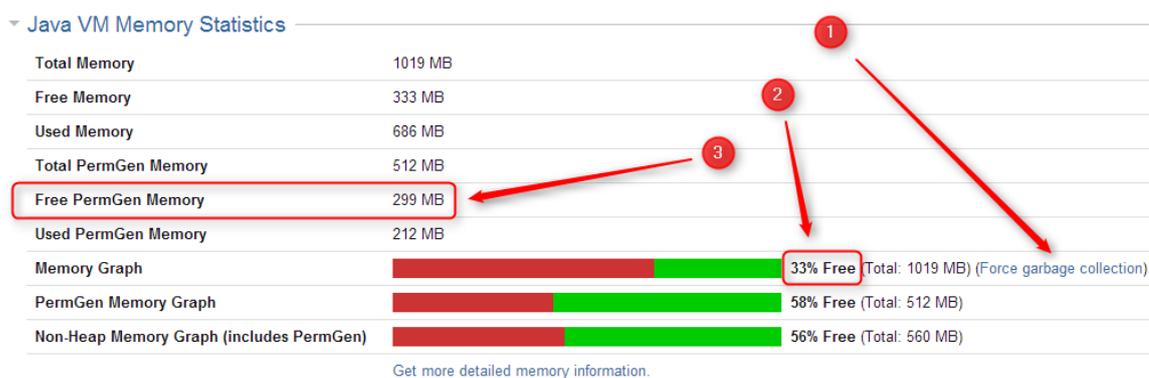
1. Open **Structure | Manage Structure**.
2. Find Global Structure and check if it has an Owner assigned.
3. If it doesn't have an owner, click **Configure** and set yourself as the owner.

### 3.1.2 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).

### 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 <code>setenv.sh / setenv.bat</code>
% 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.

## 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.

## 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.

## 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.

## 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.

## 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
<b>Total Physical Memory Required</b>	<b>8 GB</b>

### 3.1.3 Uninstalling and Reinstalling Structure

#### 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.

#### 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.

### 3.1.4 Upgrading and Downgrading

#### 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](#) 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](#).

For more specific instructions, please check the [Release Notes](#) 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](#).

## 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](#) 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](#). 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.

## 3.2 Setting Up Structure License

Unless your Jira runs on one of the [free licenses](#), 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.

### 3.2.1 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](#).
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](#) directly. In the latter case, please enter your Jira Server ID to ensure you receive a correct license.



You can also get an evaluation license from the Atlassian Marketplace. Simply go to the **Manage Add-ons** page, find the Structure add-on and click the **Try** or **Free Trial** button.



If you have installed a license you received directly from ALM Works, Manage Add-ons may show that Structure is *Unlicensed* or *Action Required*, because it's not aware of the ALM Works license. You can check the true license status on the **Administration | Structure | License Details** page — if it shows that the license is OK, you can safely ignore the status of the license in Manage Add-ons.

### 3.2.2 Licenses from ALM Works and from Atlassian

Structure supports two kinds of licenses — issued by ALM Works 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.

	License from ALM Works	License from Atlassian
Purchased at	<a href="#">ALM Works website</a>	<a href="#">Atlassian Marketplace</a>
Managed at	The license key is sent to you by email	Manage with <a href="#">Atlassian Marketplace</a>
License key looks like this:	-----BEGIN CERTIFICATE----- MIIEYTCcAkmgAwIBAgIGAT2oPFqOMA0GCSqGSIb3DQE... ... at least 20 lines of symbols ... -----END CERTIFICATE-----	AAABEA00DA... ... at least 20 lines of symbols ...
Installation Instructions	<ol style="list-style-type: none"> <li>1. If you have a license from Atlassian installed, first remove it in <b>Manage Add-ons</b>.</li> <li>2. Open <b>Administration   Structure   License Details</b>.</li> <li>3. Copy and paste the key to the <b>Install License</b> section and click <b>Install License</b>.</li> </ol>	<ol style="list-style-type: none"> <li>1. Open <b>Manage Add-ons</b>.</li> <li>2. Locate the license you want to install.</li> <li>3. Copy the license key to the <b>Install License</b> section and click <b>Install License</b>.</li> </ol>
Uninstallation Instructions	<ol style="list-style-type: none"> <li>1. Open <b>Administration   Structure   License Details</b>.</li> <li>2. See the details of the installed license and click <b>Uninstall License</b>.</li> </ol>	<ol style="list-style-type: none"> <li>1. Open <b>Manage Add-ons</b>.</li> <li>2. Locate the license you want to uninstall.</li> <li>3. Click <b>Uninstall License</b> and click <b>Uninstall License</b>.</li> </ol>
Purchasing differences	<ul style="list-style-type: none"> <li>• 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.</li> </ul>	<ul style="list-style-type: none"> <li>• Purchase <a href="#">licenses</a> from <a href="#">Atlassian Marketplace</a>.</li> </ul>

	License from ALM Works	License from
	<ul style="list-style-type: none"> <li>VAT and taxes may be handled differently from Atlassian, as our payment processors are located in the USA and Germany. ALM Works is based in the US, and for direct purchases using Wire Transfer, we do not charge VAT or any other taxes.</li> </ul>	

### 3.2.3 Purchasing a Commercial License

Structure licenses can be purchased from ALM Works, from Atlassian, or through Atlassian Solution Partners and resellers.

#### Purchasing from ALM Works

Commercial licenses from ALM Works can be purchased at <http://almworks.com/structure/purchase.html>.

To generate a license, a Jira Server ID is required. The Jira Server ID is a 16-digit code, which the Jira Administrator can look up in the Jira menu **Administration | System Info** or **Administration | Structure | License Details**.

#### Purchasing from Atlassian

You can purchase a license via Atlassian on the [Atlassian Marketplace](#).

After the purchase is completed, the license key will be available on <https://my.atlassian.com>.

#### Purchasing from Resellers or Atlassian Experts

You can purchase through a reseller of your choice. [Atlassian Solution Partners](#) can also provide you with additional services and advice.

When you purchase through a reseller, you can get either kind of license (issued by ALM Works 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.

### 3.2.4 Migrating Licenses

You can convert a license of one kind into a license of another kind. Please contact [sales@almworks.com](mailto:sales@almworks.com) for assistance.

**Next:** Select [which projects are enabled for Structure](#)

### 3.2.5 Structure License Parameters

The following parameters are displayed in the **Current License** section when you install a Structure license.

Parameter	Meaning
<b>License Type</b>	Commercial, Evaluation or other
<b>Licensee</b>	Organization authorized to use the license
<b>Serial Number</b>	A unique number assigned to the license
<b>Expires</b>	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.
<b>Maintenance Expires</b>	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.
<b>User Limit</b>	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.
<b>Server ID</b>	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.

### 3.2.6 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.

### 3.2.7 License Maintenance and Expiration

#### 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.

#### 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.

#### 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.

## 3.3 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](#)

## 3.4 Global Permissions

### 3.4.1 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](#).
- Structure is [enabled for this user](#):
  - 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.

### 3.4.2 Restricting User Access to Structure

By default, Structure is accessible to anyone who has *Browse* permission on [structure-enabled projects](#). 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](#).
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](#).

✔ 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.

✔ 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](#)).

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.

### 3.4.3 Changing Permission to Create New Structures

By default, any logged-in user with [access to Structure](#) 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](#).
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](#) to be able to create new structures.

✔ Users who have *JIRA Administrators* global permission are always allowed to create new structures.

✔ 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](#)).

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.

### 3.4.4 Changing Permission to Access Automation

By default, any user with Edit Generators [access level](#) for a structure can add and configure [generators](#). 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](#).)
  - 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](#) for that structure.



Users who have *Jira Administrators* global permission are always allowed to change generators.

### 3.4.5 Changing Permissions to Configure and Run Effectors

There are three levels of permissions for [Effectors](#):

- **Permission to Configure Effectors** - by default, anyone with [access to Automation](#) can add and configure Effectors.

- **Permission to Execute Effectors** - by default, anyone with view access to a structure 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](#).

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](#).)
  - 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.

## 3.5 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.

### 3.5.1 Initial Configuration

When Structure is installed, the defaults are configured as follows:

<b>System Default Structure</b>	None
---------------------------------	------

<b>Project Default Structure</b>	None
<b>Default Views Menu</b>	Preinstalled views: <b>Basic, Planning, Tracking, Triage, Entry</b> (on all pages)
<b>Default View</b>	<b>Basic View</b>
<b>Auto-switch (Issue Page)</b>	Structure with the displayed issue.
<b>Auto-switch (Project Page)</b>	<b>Off</b> - Show the last viewed structure.
<b>Keep structure when navigating</b>	<b>On</b> - When going from structure widget to an issue page, show the same structure.
<b>Auto-minimize Structure Panel</b>	<b>On</b> - Structure panel is initially minimized if the issue is not in the structure.

### 3.5.2 Changing Default Structure

The [default structure](#) is automatically selected when the user opens [Structure Board](#) for the first time. It is also shown when a user views the Structure widget from an issues page, if the [Auto-switch option](#) is set to *default structure*.

You can change the default structure for the entire Jira instance, as well as for specific projects.

#### 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](#) to the default structure. Users without view permissions will receive an error when the default structure should be displayed.

## 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.

### 3.5.3 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](#).

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](#).
4. Click Apply

### 3.5.4 Changing Default Options for the Issue and Project Pages

A number of options define how Structure behaves on [issue pages](#) and [project pages](#). 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.	<a href="#">Structure Options for the Issue Page</a>
Auto-switch (Project Page)	Lets you select which structure is initially displayed on Project pages.	<a href="#">Structure on the Project Page</a>
Keep Structure Selection When Navigating	If turned on, clicking an issue in the <a href="#">Structure Widget</a> 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.	<a href="#">Structure Options for the Issue Page</a>
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.	<a href="#">Structure Options for the Issue Page</a>

 If the user changes these options, those changes are preserved and are applied instead of the defaults for that specific user.

## 3.6 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.

Applications Projects Issues Manage apps User management Latest upgrade report System **Structure**

STRUCTURE ADMINISTRATION  
 Configuration  
 Defaults  
**Attributes**  
 Backup Structure  
 Restore Structure  
 Migrate Structure  
 Maintenance  
 License Details  
 Support  
 Setup Guide

## Attribute Sensitivity Settings

Attribute sensitivity settings allow you to adjust how protective Structure is when it comes to aggregating values in a structure where the user is not allowed to see some of the items (issues). This is a question of balance between securing the Jira data vs. ensuring that all users see the same aggregated result.

For example, Alice wants to see the total cost of an initiative, based on the Cost field of involved tasks. She also shares the structure with Bob. But Bob does not have access to some of the tasks in that initiative. Should Bob see the same total number as Alice does (even though he does not see all the tasks that went into that total), or should he see his own personal total based entirely on those tasks visible to him?

This is governed by whether an attribute is considered "sensitive". If Cost is not considered sensitive, the totals for Bob and Alice will be the same. Note that these settings matter only to aggregating functions. In the previous example, Bob never gets any direct access to the tasks he's not allowed to see.

### Attribute Sensitivity Mode

Mode **Standard**

**Standard** - all attributes are sensitive *except numeric and date-based*  
**Strict** - all attributes are sensitive  
**Permissive** - no attribute is considered sensitive (recommended only if all your Jira users can see all issues)

## Attribute Sensitivity Settings

### 3.6.1 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.

### 3.6.2 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.

### 3.6.3 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.

#### Attribute Sensitivity Mode

Mode Strict ▼ Change Attribute Sensitivity Mode

**Standard** - all attributes are sensitive *except numeric and date-based*  
**Strict** - all attributes are sensitive  
**Permissive** - no attribute is considered sensitive (recommended only if all your Jira users can see all issues)

#### Additional Non-Sensitive Attributes

Add Non-Sensitive Attribute

Name      Attribute Specification

Adding additional non-sensitive attributes

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.

#### Add Non-Sensitive Attribute

Spec

Attribute Story Points ▼ ⚙️

Format Number ▼

Add Cancel

Select the non-sensitive attribute



**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.

## 3.7 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

### 3.7.1 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.

### 3.7.2 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](#).

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.

### 3.7.3 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](#).



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](#) 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.

## Downgrading from Structure 5.0 or Later

The introduction of [Manual Adjustments](#) 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.

### 3.7.4 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](#).



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:

<b>Restore Structure Permissions</b>	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 <a href="#">Manage Structures</a> page.
--------------------------------------	--

<b>Restore Synchronizers</b>	<p>If selected, the synchronizers for selected structures are restored.</p> <div style="border: 1px solid #f9c77d; padding: 10px; margin-top: 10px;">  Synchronizers configuration is imported as-is, and might not make sense on a new Jira instance. After you have restored synchronizers, please visit the <a href="#">Synchronization Settings</a> page to check if the synchronizers are configured correctly. </div>
<b>Restore Structure History</b>	<p>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.</p>
<b>Restore User Favorites</b>	<p>If selected, the plugin will try to restore "favorite" marks made by users for the selected structures.</p>
<b>Restore Views</b>	<p>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.</p>
<b>Restore View Settings for Structures</b>	<p>If selected, <a href="#">view settings</a> for the selected structures will be restored.</p>
<b>Restore Configuration</b>	<p>If selected, the <a href="#">global app permissions</a> 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.</p>
<b>Restore Dark Features</b>	<p>If selected, the <a href="#">dark features settings</a> will be restored.</p>

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.

## 3.8 Automatic Structure Maintenance

### 3.8.1 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.

### 3.8.2 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.

### 3.8.3 Maintenance Tasks

<b>Backup Structure data</b>	<p>Creates a backup of the Structure database in the <code>export</code> sub-directory under Jira home.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li>• 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.</li> </ul> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> 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.</p> </div>
<b>Delete old backups</b>	<p>A backup is considered old if it is not among <math>X</math> latest backups (<math>X</math> is specified by the first parameter of this task) <b>and</b> it was made earlier than <math>Y</math> days ago (<math>Y</math> is specified by the second parameter). This task removes all such backups made by the Backup task.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li>• Always keep <math>X</math> latest backups</li> <li>• Always keep backups made during the last <math>Y</math> days</li> </ul>
<b>Optimize favorites</b>	<p>If a user marks a structure as their <b>favorite</b>, Structure will keep this mark, even if the user is later deleted from Jira, and the <b>popularity number of the structure</b> will still accounts for that user. This task removes marks made by users no longer in Jira and recounts the structure popularity.</p>
<b>Optimize structures</b>	<p>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.</p>

<b>Optimize view settings</b>	If a <a href="#">view</a> is deleted, some <a href="#">structure view settings</a> may still reference it, and a blank view named ? ( <b>Unknown View</b> ) will be shown in its place. This task removes references to deleted views.
<b>Optimize synchronizers</b>	This removes an data related to synchronizers of a deleted structure.
<b>Delete old synchronizer audit log records</b>	<p>This removes old records from the <a href="#">Synchronizer Audit Log</a>, clearing up space in the database.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li>• Keep records for the last <math>X</math> days.</li> </ul> <div style="border: 1px solid #c8e6c9; padding: 10px; margin-top: 10px;"> <p> If you set <math>X</math> to 0, the maintenance procedure will remove all records.</p> </div>
<b>Reindex change history</b>	<p>Currently does nothing.</p> <div style="border: 1px solid #bbdefb; padding: 10px; margin-top: 10px;"> <p> This task has remained as an option since Structure 2. Its purpose may be restored in a later version of Structure.</p> </div>
<b>Optimize structure perspectives</b>	<p>Removes old <a href="#">perspectives</a> that haven't been used by anyone for a certain amount of time.</p> <p>Parameters:</p> <ul style="list-style-type: none"> <li>• Delete perspectives that were not used during the last <math>X</math> days</li> </ul>
<b>Reindex structures</b>	Clears and recalculates issue-to-structure index, used to define which structures contain a specific issue. (Issues added with <a href="#">Automation</a> are not counted.)
<b>Delete old change history</b>	<p>The task removes old records from the change history. A history record is considered old if the change was made earlier than <math>X</math> days ago (<math>X</math> is specified by the first parameter) <b>and</b> it is not among the <math>Y</math> latest history records for the structure where the change was made (<math>Y</math> is specified by the second parameter).</p> <p>Parameters:</p>

- Always keep change history for the last X days
- Always keep the Y latest changes per structure

### 3.8.4 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.

## 3.9 Workflow Integration

### 3.9.1 Structure Workflow Validator

Structure Plugin adds a new [workflow transition validator](#) to JIRA. This validator blocks the transition if the issue doesn't match an [S-JQL query](#). 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](#).)
2. Select the **Issue Matches Structure Query** validator. A configuration window will open:

### Add Parameters To Validator

Add required parameters to the Validator.

Structure:  The default structure of the issue's project  
 Manually selected:   
 The one that contains the issue  
If the issue is contained in 2 or more structures, the transition will be blocked.

S-JQL Query:  ?  
The issue must match this query to pass the transition.

Validator Message:   
Optional field. Users will see this message when the issue does not match the query.

Query examples:

If the Issue Is Not Added to the Structure:  Allow transition  
 Block transition

Run as User:  Project lead  
 Specific user:   
Please enter a username.

 All operations will be performed on behalf of the specified user.

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.

### 3.9.2 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](#).



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.

## 3.10 Running Structure on Jira Data Center

Structure is a [Data Center approved app](#).

The following articles provide additional information, specific to the Data Center editions.

To learn more about the Data Center approved app program, see our [Data Center Approved Apps FAQ](#).

### 3.10.1 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.

#### 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.

#### 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).

## 3.11 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](#)

### 3.11.1 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**

### 3.11.2 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](#) and our [privacy policy](#).

### 3.11.3 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 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.

## 3.12 Structure Files

### 3.12.1 \$JIRA\_HOME/structure

Structure keeps most of its data in the `structure` sub-directory under the [JIRA home directory](#).

On JIRA Data Center, a local filesystem is used.

#### Cache files

Structure uses file system to temporarily store some of the internal runtime database, involved in Automation feature. These files may be stored in "rows0", "rows1" and similar directories.

## 3.13 Turning Off Optional Features

Some features in Structure are designed as modules and can be safely turned off. You can do so to remove unnecessary functionality, or limit the exposure of Structure plugin to the users.

 If your aim is to limit the exposure of Structure, consider restricting permissions to specific groups of users - see [Gradual Deployment](#).

 While it is easy to disable a Structure module, we don't recommend to touch any modules except those listed in this article to ensure stability of Structure and your JIRA application.

To turn off a module:

1. Open Add-on Manager by navigating to *Administration | Add-ons | Manage Add-ons*.
2. Locate 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 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 name	Effect of disabling this module
Activity Streams	Structure (structure-activity-provider)	Activity streams provider and Structure-related updates are removed from the following places: <ul style="list-style-type: none"> <li>• <a href="#">Activity Stream gadgets</a>,</li> <li>• Activity tab on the issue page,</li> <li>• Activity tab on the user page,</li> <li>• Activity tab on the project page.</li> </ul>
<a href="#">Structure on the Issue Page</a>	web-resource: Issue Page Decorator (adjustIssue)	Structure section is removed from the issue page.
<a href="#">Structure on Agile Boards</a>	web-panel: GreenHopper tab (greenhopper-tab)	Structure tab is removed from the issue details panel on the Agile board.
<a href="#">Synchronizers</a>	synchronizer:... (5 synchronizers are bundled with Structure)	Users will not be able to install synchronizers, and installed synchronizers won't run. You will need to restart the plugin to have settings make full effect. (Disable plugin, then enable plugin.)

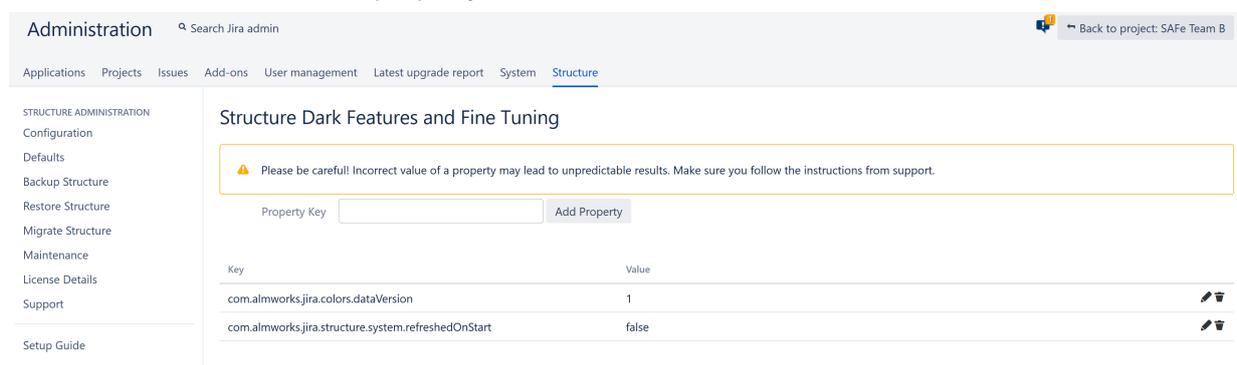
## 3.14 Advanced Configuration and Dark Features

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.

### 3.14.1 Setting Application Properties with the Structure Dark Features and Fine Tuning Interface

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.



The screenshot shows the Jira Administration interface for 'Structure Dark Features and Fine Tuning'. At the top, there is a search bar and a 'Back to project: SAFe Team B' link. Below the navigation tabs, a sidebar lists various administration options. The main content area features a warning message: 'Please be careful! Incorrect value of a property may lead to unpredictable results. Make sure you follow the instructions from support.' Below this is a form with a 'Property Key' input field and an 'Add Property' button. A table below the form lists existing properties with their keys and values, and includes edit and delete icons for each row.

Key	Value	
com.almworks.jira.colors.dataVersion	1	 
com.almworks.jira.structure.system.refreshedOnStart	false	 

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](https://YOUR_JIRA_ADDRESS/secure/admin/StructureDarkFeatures.jspa)

### Guidelines for Adding/Removing Property and Values

- 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, its 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.

### 3.14.2 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.

## Setting System Properties 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](#). You will need to restart Jira for the properties to take effect.

## Setting System Properties with Script Runner

You can also set system properties using the [Script Runner](#) 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, 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.

### 3.14.3 Structure size limit

Property	Default	I
<code>com.almworks.jira.structure.AOBasedStructureManager.forestSizeLimit</code>	100000	-   

Property	Default	Explanation

### 3.14.4 Structure Automation limits

Property	Default	Explanation
<code>structure.gfs.generationTimeHardLimit</code>	600	The maximum amount of time that can be spent for Structure generation (in seconds).

### 3.14.5 Automation Defaults

Property	Default	Explanation
<code>structure.generator.defaults.disableUpdates</code>	false	When adding generators: <ul style="list-style-type: none"> <li>• If set to "false" (default) - the "allow changes" box is initially checked.</li> <li>• If set to "true" - the "allow changes" box is initially unchecked.</li> </ul>

### 3.14.6 Manual adjustments

Property	Default	Explanation
<code>structure.gfs.manualAdjustments.enable</code>	true	Setting this property to false will disable manual adjustments for the entire Jira Instance. All adjustment-related UI elements

Property	Default	Explanation
		and controls will disappear. Existing manual adjustments will be kept in the database, but will not be applied.
<code>structure.gfs.manualAdjustments.maxAdjustmentsPerStructure</code>	2000	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.
<code>structure.gfs.manualAdjustments.maxAdjustmentsPerAction</code>	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.

### 3.14.7 Hidden Issue Links

Property	Default	Explanation
<code>structure.feature.hiddenLinks.enabled</code>	false	Set to true to enable support for hidden issue links.

### 3.14.8 Index Consistency Checks

Property	Default	Explanation
<code>structure.indexConsistencyChecker.disabled</code>	false	Set to true to disable periodical checks of Lucene index consistency.

### 3.14.9 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" or "+Next Issue" was pressed.

<b>System property</b>	<code>structure.feature.altInitialValuesInDialog</code>
<b>Options to add in setenv.sh / setenv.bat</b>	<code>-Dstructure.feature.altInitialValuesInDialog=true</code>
<b>Internal feature name</b>	<code>altInitialValuesInDialog</code>
<b>Introduced in version</b>	2.11.0

## Synchronizers

[Synchronization](#) lets you keep Structure issue hierarchy in sync with some other issue properties. This dark feature is disabled by default.

Property	Default	Explanation
<code>structure.feature.synchronizers.enabled</code>	<code>false</code>	Set to true to enable Synchronizers within Structure.

## Synchronizer Cycle Guard

The [cycle guard](#) 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
<code>structure.sync.guard.disable</code>	<code>false</code>	Set to true to disable the cycle guard. Conflicting

Property	Default	Explanation
		synchronizers will not be prevented from running forever. <b>Not recommended.</b>
<code>structure.sync.guard.maxAutosyncsWithoutUserChanges</code>	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.
<code>structure.sync.guard.stop.disable</code>	false	If true, conflicting synchronizers will not be disabled automatically. The cycling may repeat after a user-generated change.
<code>structure.sync.guard.email.owner.disable</code>	false	If true, the cycle guard will never send e-mail notifications to synchronizer owners.

Property	Default	Explanation
<code>structure.sync.guard.email.admin.disable</code>	false	If true, the cycle guard will never send e-mail notifications to Jira administrators.
<code>structure.sync.guard.email.admin.cycles</code>	10	<p>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.</p> <p>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.</p>

### Resolved icon(green tick)

Property	Default	Explanation
<code>structure.doneAttribute.byResolution</code>	false	false - the "Resolved icon" shown when the Resolution field of an issue is non-empty

Property	Default	Explanation
		true - the "Resolved icon" is shown when the issue is in a Done status category (StatusCategory.COMplete)

## Time in Status - Refresh Period

Property	Default	Explanation
<code>structure.timeinstatus.refreshPeriod</code>	3600000	<p>Sets update period for Time in Status columns. Value is in milliseconds.</p> <p>Time in 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.</p>

## Link-related changes made by Generators or Synchronizers

By default, when a generator (see [Automation](#)) 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.

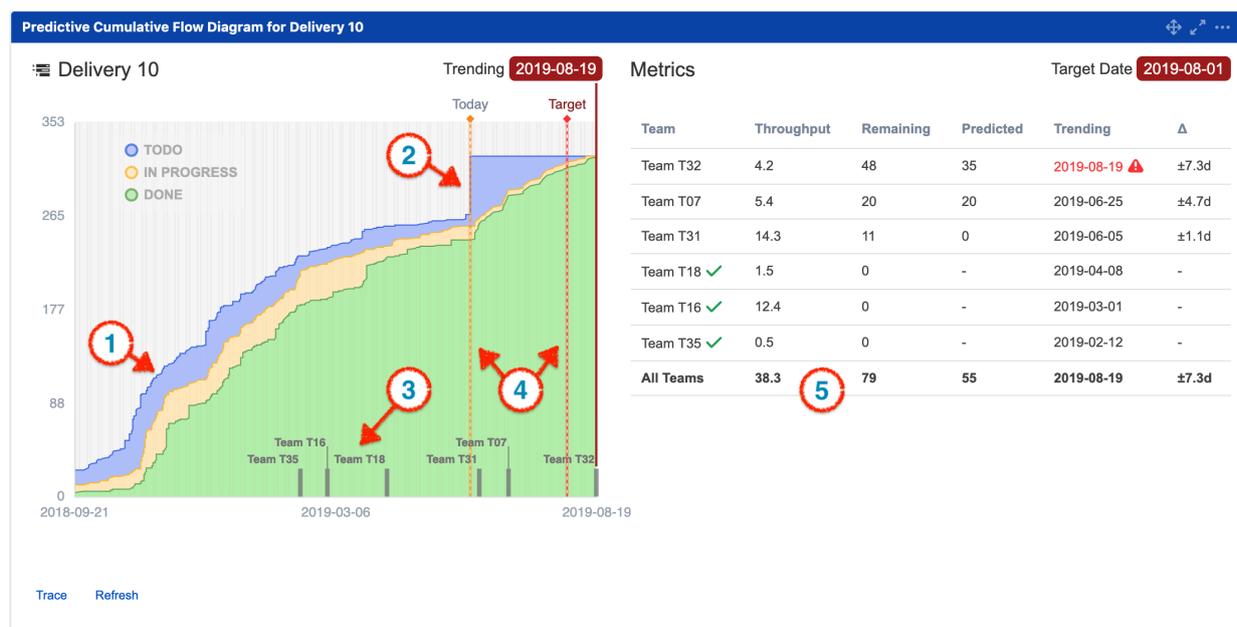
This setup prevents link changes from negatively impacting system performance (a particular risk when bulk changes are made to issue links); however, there may be times when these types of changes need to be indexed immediately and/or appear in an issue's history. To enable this, simply update the following properties.

Property	Default	Explanation
<code>structure.bulkLinkProcessor.useLinkManager</code>	false	true - enables the immediate re-indexing after every change, writes changes in the history and creates a

Property	Default	Explanation
		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.

### 3.14.10 Structured Predictions

The Structured Predictions gadget uses the issue and issue workflow data in Jira and Structure to create Predictive Cumulative Flow Diagrams, which allow organizations to visualize a project's progress and expected completion dates across teams, even before all scope data is available in Jira.



Using historical team performance data, Structured Predictions is designed to answer the fundamental questions of business leadership: “when will I see value from my investment?” and “what can we do now to improve outcomes later?” Much of the work of project management is found in preparing answers to these questions.

Structured Predictions is designed to provide support to program managers, planners, and teams accountable for answering these questions. It makes the answers consistent, measurable, comparable, and easy to get.



Structured Predictions is what we refer to as a [Dark Feature](#). It ships with Structure as part of your normal Structure license, but you need to enter a special code to enable it. To receive the code, please contact our Solutions Team at [solutions@almworks.com](mailto:solutions@almworks.com).

## Learn more about Structured Predictions

### Configuring Structured Predictions

The following guide describes how to configure Structure and Jira to get the most out of Structured Predictions.

#### Prerequisites

You must have Structure 5.5 or higher installed.

### Configuring Structure Data Sources

Structured Predictions uses [Structure for Jira](#) from ALM Works to define data sources used for predictions. Configuration options allow for some flexibility in how the Structure data sources are configured, but some guidelines will make it easier to align your data with the Structured Predictions configuration options.

There are two primary data sets Structured Predictions needs to be able to identify:

- 1. The issues that serve as estimates (typically Epics)** - these represent the more abstract requirements that will group the scope level into "kinds of work" a team will need to do in order to satisfy the requirements for a delivery.
- 2. The issues that identify scope (typically Stories)** - these represents the specific units of work an engineer will deliver in a sprint, or as part of the normal process flow if you are using Kanban or some other practice.

A structure that is configured to show Epics as estimates and Stories under those epics to show refined scope is in good shape to feed directly into Structured Predictions. A structure configured this way will look something like the figure below.

Key	Summary
DELIVERY-1	Storefront Integration with Zipmod
T06-2	Transmission Module
T06-3	Submit button

## Configuring Structured Predictions

When you create a new Structured Predictions gadget, you will specify a number of configuration elements, including how to identify and interpret the information discussed above.

**Predictive Cumulative Flow Diagram for Delivery 10**

Scope Structure\*

Training Structure\*

Training Period\*

Estimation Level\*

Scope Level\*

Team Attribute\*

Target Date

**Advanced Options**

When Added?\*

When Work Started?\*  ⚙️

When Completed?\*  ⚙️

Configuration Options for the Predictive CFD

### 1. Scope Structure

Select the structure that contains the work you want to track.

### 2. Training Structure

Select a structure that contains one or more completed deliveries. This structure will be used to train the Structured Predictions algorithms as to how the teams working in the Scope Structure have performed in the past. This structure should contain work representation for all teams working in the Scope Structure. It can be the same structure as the scope structure, but if the scope structure does not have any data or only has a little data the results will be highly variable.

### 3. Training Period

Select whether to use all historical data or just the last few weeks worth of data to train the algorithms. If only the past few weeks are desired, make sure to set the Training Structure equal to the Scope Structure. This will show how the teams have been performing in recent history and how that performance translates into target dates for the delivery.

### 4. Estimation Level

Select what issue types your organization uses to specify a work definition that is larger than an Agile Story. This is usually an Epic, but if your organization uses other issue types, make sure to include them here.

#### 5. Scope Level

Select what issue types your organization uses to indicate discrete work a developer can deliver. On Agile teams, this is usually a Story and a Bug issue type. If your teams use other issue types, indicate them here.

#### 6. Team Attributes

Indicate how Structured Predictions should identify team backlogs in the data sets defined by the Scope and Training Structures. For example, if each of your teams has its own project, then indicate "Project". If you use a custom field, indicate which one. If you use the built-in Jira component field, select that. If you have a more complex method, you can use a Structure Formula to isolate the team backlogs in the Scope and Training Structure data sets by selecting "Formula" and specifying the proper formula code snippet.

#### 7. Target Date

Specify what date the delivery is expected to be delivered.



Structured Predictions is what we refer to as a **Dark Feature**. It ships with Structure as part of your normal Structure license, but you need to enter a special code to enable it. To receive the code, please contact our Solutions Team at [solutions@almworks.com](mailto:solutions@almworks.com).

## Using Structured Predictions

### Reading the Structured Prediction Gadget

Structured Predictions generates a Predictive Cumulative Flow Diagram and provides specific metrics for each team.



The following sections explain each element of the Structured Predictions gadget.

## 1. Historical Cumulative Flow Diagram

This section of the graph shows the historical CFD of past workflow transitions and ticket inventory to the current day. The points on the graph show the date at which issues were in a specific workflow state:

- Blue graph area indicates the work that is still todo (in the backlog)
- Yellow indicates the work that is in progress
- Green indicates the work that is done

By plotting all of the issue transitions for the project, the graph shows a historical representation of what the project looked like at a given date in history. By assessing the slope of each workflow state, the rate at which work is being created and completed for a project can be easily seen.

## 2. Predicted Stories

The steep cliff in the "todo" area of the graph at the line labeled "Today" is the result of the Structured Predictions algorithm making a predictions as to the number of additional stories each team contributing to the delivery will create at some future point in the project. The predicted stories are added to the known project scope (the scope that is defined in Jira) at the current date.

### 3. Estimated Completion Dates

Each team's completion date is predicted by running the team's performance model (based on the training structure assigned in the [configuration](#)) against the remaining scope defined in Jira and the additional scope predicted by the algorithm for that team. Hovering over the team label pin shows some metrics for each team.

### 4. Today and Target Lines

The Today line, quite simply, is drawn at the current date. It represents the transition point of the graph from a traditional historical cumulative flow diagram to a predictive cumulative flow diagram. The portion of the graph to the right of the Today line is drawn from the simulated future for each team working on the delivery.

The Target date is provided to the gadget as part of the [configuration](#). Teams that are predicted to finish after the target date are flagged in red in the metrics table to the right of the graph.

### 5. Team Metrics

The team metrics breaks down the key stats by team. There are six columns of data for each team.

1. Team - is the name of the team as determined by the "Team Attribute" setting in the Predictive CFD gadget configuration screen.
2. Throughput - is the average throughput for the team measured in items of scope per week. What issues represent scope is configurable. The default is stories and tasks. For example, "1.0" indicates that a team generally gets one story per week done. A value of "3.30" would indicate that every week a team gets a little over three items of scope done.
3. Remaining - indicates how much scope remains out of the total scope, actual plus simulated, for each team.
4. Predicted - reveals how much of that scope, column 3, is from the predictive algorithm. Predicted scope is scope that is not yet represented in Jira but is predicted to be created by the team at some point in the future. For example, a value of 48 in column 3 and 35 in column 4 tells you that while there are only 13 remaining issues defined in Jira ( $48 - 35 = 13$ ), there are likely to be an additional 35 created in the future for a total of 48 remaining items of scope for the team.
5. Trending - shows the predicted completion date for the team. This is when the team will likely complete all known and predicted scope.
6. Delta - shows the standard deviation for each team's throughput. This is the standard deviation for the underlying data used to determine the throughput. Teams with low deviation have a more consistent focus on the project work, and correspondingly their

predictions will be more trustworthy. Teams with higher deviations may require more attention to make sure they are focused when they need to be. They may be juggling multiple priority streams, or they may have a less predictable or mature process. Whatever the cause of the deviations, attention to the team is advisable.

## Using Predictions to Set Target Dates

The Structured Predictions graph is a great way to start a conversation with the teams on a project about what dates they think they can hit, given the particulars of the project and the circumstances within the organization:

- Talk about the output from Predictions with the various teams.
- Let them know what project data you are using for the predictions.
- Ask them if the dates look good, are too aggressive, or if they think they can beat them. Ask them why they believe what they do.
- Record the key points from the dialog, along with the agreed date. These key points will provide context once the project gets further along and decisions need to be made as the project responds to change.

## Including Predictions in Status Reports

Taking a screen shot of Structured Predictions is a quick way to get completion trends into a status report. A weekly report with a screen shot will reveal trends to stakeholders as well as validate the impact of decisions that were made and expected to impact project performance.

 Structured Predictions is what we refer to as a [Dark Feature](#). It ships with Structure as part of your normal Structure license, but you need to enter a special code to enable it. To receive the code, please contact our Solutions Team at [solutions@almworks.com](mailto:solutions@almworks.com).

## Structured Predictions FAQ

### What are the requirements for Structured Predictions?

Structured Predictions requires Structure for Jira v5.5 or above. In addition, to receive the best results, you should refer to our [Structured Predictions Configuration Guide](#).

### How do I get Structured Predictions?

Structured Predictions is what we refer to as a [Dark Feature](#). It ships with Structure as part of your normal Structure license, but you need to enter a special code to enable it. To receive the code, please contact our Solutions Team at [solutions@almworks.com](mailto:solutions@almworks.com).

## How do I use Structured Predictions?

While the Structured Predictions feature is free as part of your Structure license, it does require some configuration and training in order to be effective. Please contact our Solutions Team via [solutions@almworks.com](mailto:solutions@almworks.com) for details about enabling Structured Predictions for your environment.

## How long does it take to install Structured Predictions?

Structure Predictions is deployed by the ALM Work's Solutions Team. The deployment includes:

- Initial assessment
- Installation of Structured Predictions itself
- Training for the key stakeholders that will be using it
- A presentation to a larger audience about what the product is and how it can be used to improve planning, execution, and delivery.

The duration depends on your current Jira configuration, development process, and other factors. Please contact our Solutions Team at [solutions@almworks.com](mailto:solutions@almworks.com) to set up a meeting to review your specific situation.

## How much does the solutions configuration support cost

Pricing for Structured Predictions implementation services vary, depending on your needs. Please contact our Solutions Team at [solutions@almworks.com](mailto:solutions@almworks.com) to set up a meeting to review your specific situation.

### 3.14.11 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](#) 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](#) places Jira sub-tasks under their parent issues in the structure.
- [Links Synchronizer](#) 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](#) 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](#) 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](#) 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](#) or [Import](#), or when you run a [Resync](#). 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.

## Import Synchronization



### Dark Feature - Synchronizers are hidden by default

We recommend using [Automation](#) 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](#).

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](#) to add All Open Issues to a structure (or the results of whatever Saved Filter you have), or you could use the [Agile Synchronizer](#) 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](#) 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](#).
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** and **Export** are actually a one-time **Resync**. 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!

## Export Synchronization

### **Dark Feature - Synchronizers are hidden by default**

We recommend using [Automation](#) 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](#).

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](#) 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](#) 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](#).
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** and **Export** are actually a one-time **Resync**. 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!

## Installing a Synchronizer

 **Dark Feature - Synchronizers are hidden by default**

We recommend using [Automation](#) 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](#).

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](#) or [export](#)).

Here are a couple of examples of how that might work:

- With a [Links Synchronizer](#) 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](#) 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](#) 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](#).
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](#) 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** and **Export** are actually a one-time **Resync**. 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!

## Modifying Synchronizer



### Dark Feature - Synchronizers are hidden by default

We recommend using **Automation** 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](#).

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** 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** 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.

## Removing Synchronizer

### **Dark Feature - Synchronizers are hidden by default**

We recommend using **Automation** 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](#).

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.

## Turning Synchronizer On and Off



### Dark Feature - Synchronizers are hidden by default

We recommend using [Automation](#) 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](#).

A synchronizer can be in one of the following states:

- **Disabled** - the background incremental synchronization is not running. You can run [resync](#) 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](#), they must be enabled, typically with a [resync](#).

## 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](#) 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.

## Running Resync

### **Dark Feature - Synchronizers are hidden by default**

We recommend using [Automation](#) 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](#).

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](#).
- *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](#).

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.

## Synchronization and Permissions

 **Dark Feature - Synchronizers are hidden by default**

We recommend using [Automation](#) 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](#).

**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](#) 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](#) 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](#).

By default, any logged-in user with Control [access level](#) for a structure can manage that structure's [Synchronizers](#). 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](#).
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.

## Protection from Synchronizer Cycles



### Dark Feature - Synchronizers are hidden by default

We recommend using [Automation](#) 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](#).

It is possible to accidentally create a pair of synchronizers that would contradict each other. For example, a [sub-tasks synchronizer](#) can be configured to put a sub-task under an issue, while a [links synchronizer](#) 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](#).

## Bundled Synchronizers

### **Dark Feature - Synchronizers are hidden by default**

We recommend using [Automation](#) 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](#).

There are several bundled synchronizers included with Structure. Other synchronizers can be provided by other Jira add-ons.

 Synchronizers are now a dark feature, meaning they are hidden from users by default. To enable synchronizers, use the [Structure Dark Features and Fine Tuning Interface](#).

## Sub-Tasks Synchronizer

### **Dark Feature - Synchronizers are hidden by default**

We recommend using [Automation](#) 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](#).

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](#)).

## 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](#) 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](#).

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

<b>Filter</b>	<p>This is where you determine the Saved Filter or JQL Query to sync with. Click <b>Select</b> to choose a saved filter or switch to <b>JQL query</b> and enter the JQL.</p> <div style="border: 1px solid #f9c775; padding: 10px; margin-top: 10px;"> <p> 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 <code>updatedAt &gt; startOfMonth()</code>, 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 <a href="#">Resync</a> or use scheduled synchronization.</p> </div>
<b>Add</b>	<p>Turns on <b>Add Mode</b>: the synchronizer will make sure that all issues from the filter's result are present in the structure.</p>
<b>Place added issue at the top level</b>	<p>If this is selected, the newly discovered issues from the filter result are placed at the top level, at the end of the structure.</p>
<b>Place added issue as a sub-issue of ...</b>	<p>Allows you to select 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.</p> <div style="border: 1px solid #f9c775; padding: 10px; margin-top: 10px;"> <p> If the specified issue is not present in the structure, newly discovered issues will not be added to the structure.</p> </div>
<b>Allow move</b>	<p>This option is only available if you select the "Place added issue as a sub-issue of..." option.</p>

	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.
<b>Remove</b>	<p>Turns on <b>Remove Mode</b>: 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.</p> <div style="border: 1px solid #f08080; padding: 10px; margin-top: 10px;"> <p> <b>CAREFUL!</b> 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 <a href="#">Protection from Synchronizer Cycles</a> for more details.</p> </div>
<b>Remove only from where added issues are placed</b>	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](#)).

 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](#).

## 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](#) 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](#).

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](#), which will add Open Stories to the top level, and the [Links synchroniser](#), 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](#), 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](#) 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](#).

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](#)). Incremental synchronization watches both structure changes and issue link changes and applies those change to the other side (unless the [Reverse contradicting changes](#) option is specified).

✔ No matter how synchronizers are configured, they will only affect issues from the projects that are [enabled for synchronization](#).

✔ 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](#), 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](#) describes how to add JIRA startup properties.

2. Restart Jira

## Agile Synchronizer



### Dark Feature - Synchronizers are hidden by default

We recommend using [Automation](#) 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](#).

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

<b>Synchronize</b>	Choose the mode of operation: <ul style="list-style-type: none"> <li>• Use Agile Board configuration</li> <li>• Use custom projects and fields configuration</li> </ul>
	<b><i>Agile Board mode parameters:</i></b>
<b>Agile Board</b>	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.
<b>Synchronize Rank</b>	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.
<b>Synchronize Epics</b>	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.
<b>Auto-add sub-tasks</b>	If checked, all sub-tasks of an issue will be added to the structure as well and positioned under the parent issue.
	<b><i>Custom projects and fields parameters:</i></b>
<b>Projects</b>	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.
<b>Rank Field</b>	The field of type "Rank" that holds the rank (backlog order) for the selected Project. If you do not wish to synchronize rank, select <i>Don't synchronize</i> .
<b>Synchronize Epics</b>	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.

<b>Auto-add Subtasks</b>	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](#)). 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](#) 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.

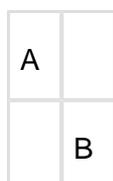
To add issues to the structure without breaking the existing backlog order:

- 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:



	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](#) 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](#).

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

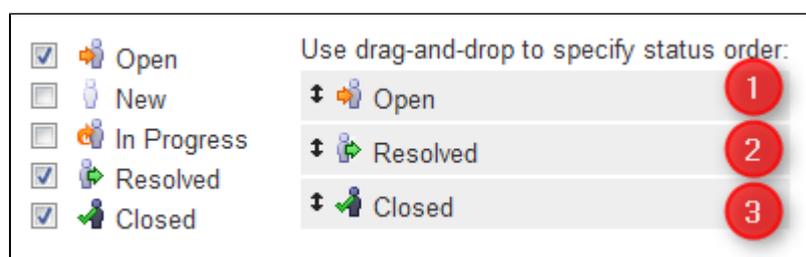
<b>Enabled Projects</b>	Only issues belonging to the selected projects are <b>changed</b> . It does not matter what project sub-issues belong to, as long as their parent belongs to the enabled project — every sub-issue counts with its status.
<b>Enabled Issue Types</b>	Determines which types of issues can be changed by the synchronizer. As with the enabled projects, only the parent issue type is checked.
<b>Statuses Rolled Up</b>	The selection and order of statuses that are used to calculate the parent issue status. Parent issue status is set to the <i>earliest</i> status among its sub-issues. If a sub-issue has a status not selected in this parameter, the parent issue is not changed.
<b>Allowed Transitions</b>	For every status, you can select which transitions the synchronizer can make to move an issue to that status.
<b>Resolution</b>	The value the <i>Resolution</i> 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:



**i** 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.

Key	Summary	Status
TS-1	Parent Issue	OPEN
✓ TS-2	Sub-issue 1	RESOLVED
TS-3	Sub-issue 2	OPEN
TS-4	Sub-sub-issue 2.1	OPEN
✓ TS-5	Sub-sub-issue 2.2	RESOLVED
✓ TS-6	Sub-issue 3	CLOSED
✓ TS-7	Sub-sub-issue 3.1	CLOSED

Issue status is set to the earliest status of its direct sub-issues

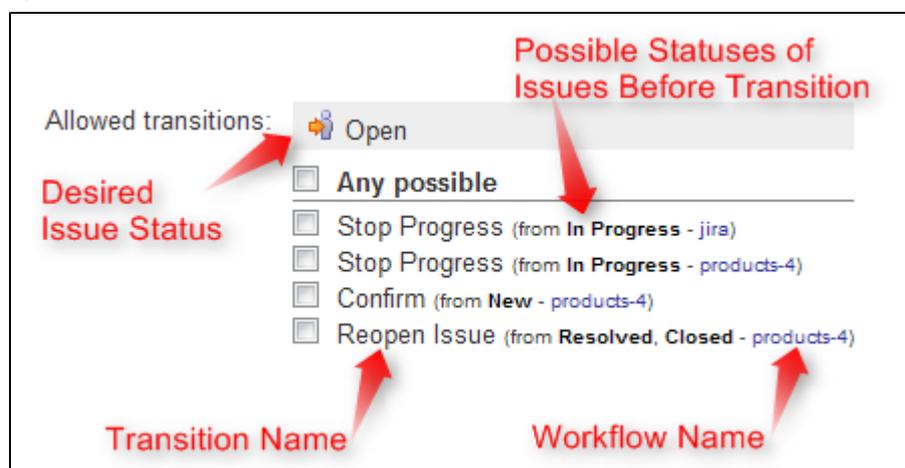
On the screenshot above:

- **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.

**i** 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.

**i** 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.

## Undo Synchronizer Actions

### **Dark Feature - Synchronizers are hidden by default**

We recommend using [Automation](#) 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](#).

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**.

View [Undo](#)

Selected log entries

[Expand all actions](#) | [Collapse all actions](#)

<input checked="" type="checkbox"/>	Time	Structure	SIID	Synchronizer	Entry description (Expand all, Collapse all)
<input checked="" type="checkbox"/>	2018-08-21 16:08:58.435 +0400	<a href="#">#199 Synchronizer Test</a>	3	Issue Links	+ <b>action:</b> sync ... <b>result:</b> success

**Actions (1): collapse, undo**  
**type:** createLink, **source:** (key: STMA-24, id: 13000), **destination:** (key: STMA-3, id: 12718), **linkTypeId:** 10000

On the next screen, review and confirm your selection by clicking **Undo**.

View [Undo](#)

Selected actions

**type:** createLink, **source:** (key: STMA-24, id: 13000), **destination:** (key: STMA-3, id: 12718), **linkTypeId:** 10000

[Undo](#)

## Copying Synchronizers



### Dark Feature - Synchronizers are hidden by default

We recommend using [Automation](#) 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](#).

When [copying a structure](#) 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.

## Copying Parameters

Synchronizers 2  
Sub-Tasks (Sub-task, Technical task)  
Links (Blocks)

Copy  No  
Synchronizers?  Yes

Synchronizer ownership:  
 Leave as is  
 Make me (admin) the new owner

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](#).

## 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

### 3.14.12 Structure 6.0 EAP

## 3.15 System Requirements

### 3.15.1 Atlassian Platform

<b>Jira Versions Supported</b>	7.13 – 8.9 (by the latest version) See also: <a href="#">Platforms supported by Jira</a>
<b>Jira Editions Supported</b>	Jira Core, Jira Software, Jira Service Desk
<b>Jira Data Center</b>	Supported See Server Requirements below
<b>Confluence Versions (Structure.Pages)</b>	6.13 – 7.5

### 3.15.2 Databases

Databases used by Jira are also supported by Structure.

### 3.15.3 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	
Internet Explorer	11	8, 9, 10
Safari	All recent versions on OS X	Safari for Windows is not supported.
Edge	All recent versions	
Other browsers	Unsupported, but may work	

### 3.15.4 Server Requirements

- At least 100MB of free disk space is needed on the server. See [Structure Files Location](#) 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](#) for details.
- Jira process must have read/write permissions to the Jira (local) home directory to create the `structure` sub-directory automatically.

### 3.15.5 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.

## 3.16 Confluence Gadget - Admin Configuration

The following articles will assist admins with enabling and configuring the Structure gadget within Confluence.

### 3.16.1 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](#)

- a. Use **OAuth Authentication** to let the Confluence page viewer authenticate separately with Jira. (Preferred)

Documentation: [Configuring OAuth for an Application Link](#)

- 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](#)



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](#)

3. Check the sample page to confirm whether you can include the Structure macro and get data from Jira.

### 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](#).

### 3.16.2 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](#)):

1. Copy `cors-filter-2.4.jar`, `java-property-utils-1.9.1.jar` from [CORS docs](#) 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

### 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:

```
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;
}
```

## 3.17 Best Practices

We have collected several guidelines for common situations.

If you have your own best practice to suggest, please [let us know!](#)

### 3.17.1 Best Practices and Performance Considerations

These documents are intended for anyone who is overseeing an implementation of the Structure App in an organization. It contains:

- suggestions for an effective Structure roll-out
- performance considerations
- general recommendations you can use to build more detailed guidelines for the end users, based on their specific use cases
- ideas for troubleshooting

Please refer to the sections below for specific information:

#### Enabling Structure

When introducing Structure in an organization, it may make sense to limit the use of it to a pilot project or some specific group of users. This can be done via the Structure Administration page (Administration | Structure).

On this page, you can configure the following:

- [Projects with Structure](#). You can enable structure for all projects, or you can limit its exposure to a number of specific projects. If Structure is not enabled for a certain project, issues from this project cannot be added to structures, and users who only have access to such projects will not see Structure at all.



If you create a new project in Jira that should be using Structure, don't forget to add it to the list of enabled projects (unless you have selected the All Projects option).

- [Users of Structure](#). To define which users have access to Structure (on top of any restrictions by projects), you can use Jira Groups and Project Roles.



When Structure is enabled for **anyone**, even anonymous visitors will have access to Structure. This means that even if they don't have access to projects and will not see any issues in a structure, they will still see the structure name and may see the [Automation](#) settings. To make Structure accessible to only logged in users, restrict access to the **jira-users** group.

- [Users who can create structures](#). By default, any logged-in user with [access to Structure](#) can create new structures of their own. Using this setting, you can restrict this permission.
- [Permission to create Synchronizers](#). By default, any logged-in user with Control [access level](#) for a particular structure can manage [Synchronizers](#) of that structure. Using this setting you can restrict this permission.



Synchronizers were a mechanism for building self-updating structures in older versions of Structure (pre 3.0). In version 3.0, [Automation](#) was introduced to replace them, and synchronizers are gradually being faded out. Starting with version 5.0, synchronizer functionality is no longer visible if there were no synchronizers installed on the instance previously. To access synchronizers, they must first be enabled using the [Structure Dark Features and Fine Tuning Interface](#).

If you do see synchronizers on your instance, it might be best to restrict this permission to admins only, to prevent users from creating new synchronizers, and encourage the use of [Automation](#) instead.

- [Permission to manage Automation settings](#). By default, any user with Edit Generators [access level](#) for a structure can add and configure [generators](#). Unlike synchronizers, generators cannot cause any unexpected changes to Jira data (see [Automation and Synchronizers](#)), so from this point of view, it's safe to let users work with them. Automation can, however, affect the server performance if they are used to build very large structures (see [General Performance Considerations](#)). This is especially important in large instances. In version 4.6 we introduced the [Automation Timeout](#) mechanism, which will stop a generation process if it is taking longer than a certain period time; however, it may still be a good idea to introduce the Automation feature gradually. A good practice could be to identify a set of power users to start using Automation immediately, and then organize some use-case-based training or create instructions for less experienced users.

## Building Efficient Structures With Automation

The following guides provide an overview of how Automation works and best practices for using it efficiently within structures.

### Basic Terms and Principles

Automation is the mechanism that allows users to build "dynamic" structures. Such structures show Jira issues arranged into hierarchies, based on the relations between issues, such as Issue Links, Epic Links, Sub-Task, grouping by Jira fields, etc. The hierarchies are automatically updated when the underlying Jira data changes, so they always provide real-time data representation. It is also possible to change the underlying data and relationships by adjusting the structure hierarchy.

Automated structures are built using generators - rules for how the structures are created and organized. Different types of rules produce different results. They allow you to:

- Insert - add issues into the structure. Insert generators are usually used to pull in an initial set of issues, from which the rest of the structure is built.
- Filter - hide some issues from the structure or from certain levels in a structure.
- Sort - sort issues by some field on all levels or on a particular level.
- Group - organize issues into groups based on the values of some field.
- Extend - add additional issues to a structure based on links to the issues already in the structure.

By combining different generators together, you can build custom structures for any business purpose. For some useful examples, see [Common Approaches to Building Structures](#).

## Generator Scope

Generators can be added under any "static" part of the structure. There are several common options here:

1. You can add them directly under the root item (contains the structure name, which becomes visible once you switch to Automation editing mode). In this case, their scope will be the entire structure. This is the best option if the entire structure is based on the same set of generators.
2. You can start by adding some folders or issues into your structure manually (dragging in existing ones or creating new ones). Then you can add generators under these folders or issues - in this case the generators will only affect the part of the structure built under each particular folder or issue. This is useful if you want to combine several sets of issues in one structure, but you want to organize them differently. For example, this allows you to include data from several teams, even if they organize their issues differently. In this case, you can visualize the work of each team under a different folder and then see the progress, totals or formula results for the entire hierarchy across all teams.

## Order of Execution

The generators added under the same parent are always executed in the following order, regardless of the order in which you add or arrange them:

1. Insert generators
2. Extend generators
3. Filter generators
4. Group generators
5. Sort generators

If there are multiple generators of the same type, they are run in the order they are listed, from top to bottom.

- This does not affect the results of Insert, Extend or Filter generators.
- This does affect the results of Group and Sort generators. With multiple Group generators, items will be grouped first by the top-most Group generator, then by the second highest, etc. The same is true with Sort - the order of the Sort generators defines which field the structure is sorted by first, second, and so on.

To change the order generators of the same-type are run, simply move them up or down in the structure.

## Decision Panel

Whenever you perform an action in Structure that is either not supported by the configured generators or requires your attention, you will see a decision panel at the bottom of the screen with a description of the problem and suggested actions. There are several common situations you may see when working with Automation:

- *Cannot move a non-issue here.* Such messages are shown if you try to add a new generator or move an existing generator under an issue that was added by another generator. Generators can only be added under static content. You can click Acknowledge to undo the move or creation, or drag it under the correct parent (a static issue or folder or the root).
- *What would you like to do?* You will see such messages if you are moving an issue under a parent which falls under the scope of several generators and it's not clear which of them should be used. For example, if you move an issue under a new parent that is in the scope of a Linked Issues Extender and a Stories Under Epics Extender - You will need to decide whether you want to change the issue link or the epic link. To minimize such requests, make sure you set the level setting for different generators when it makes sense, so that different generators work on different levels.
- You may see other decision panels when you try to make a change to the structure which has no corresponding generator or is not allowed by Jira. For example, trying to reorder issues when there is no sorter, trying to put an epic under another epic when it's in the scope of a Stories Under Epics Extender (since epics cannot have epic links), etc.

## Manual Adjustments

For situations when you do want to adjust your structure in a way that is not supported by the generators (override the generator rules), you can enable [manual adjustments](#).

## Common Approaches to Building Structures

Here are some common approaches you can use to visualize your work breakdown structures.

### Top-Down Approach

#### Common use cases:

Such approach works best when you need to visualize the complete hierarchy of issues, starting from the most high-level elements all the way down. For example, a work breakdown structure for multiple projects, with some high-level initiatives on top of tasks that implement those initiatives, which are then broken down into smaller tasks. Such an overview works well

for tracking progress across a large project, where it's important to roll-up data such as estimates, time spent and progress from low levels all the way to the top. Once built, you can then drill down to see the particular details - for example, find the tasks which are hindering the progress of an initiative.

### **Performance considerations:**

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.

It's best to keep these structures visible to the appropriate users only - for example, project or program managers. It's not recommended to build such structures for individual teams, which are interested only in a sub set of the structure (we'll cover what to do for them below) .

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's 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 same idea should be applied to the sub-tasks extender.

If you have several levels linked with a particular link type, you can set the levels as a range. Please note, that the **To** level is the last level that should be scanned for issues to be pulled in, not the level that is pulled in. For example, if you have a Theme -> Initiative -> Epic hierarchy in which three levels are connected with the same link type, you want to set the levels to: *1 to 2*. This means the generator will scan the top level (themes) and will pull in the second level (initiatives). Then it will scan the second level and pull in the third level (epics), and then it will stop. There is no need to scan the third level, because there is nothing linked to it with this link type.

**Limitations:**

Since such a structure is built top-down, we will only see issues that have an incoming link. This may not be convenient when we want to visualize the entire set of issues in the form of a hierarchy. For example, we may want to visualize issues from a number of Scrum Boards. In this case, stories which do not have an epic defined will not be shown in such a structure.

There are several ways to resolve this. One approach is to take the [Visualizing the Entire Scope](#) approach.

Another approach is to use the secondary panel to identify the missing issues. Open the secondary panel, select the JQL option and specify the JQL which describes the scope you want to see in a structure. Click the Diff button in the toolbar (       ) to hide issues which are already present in your structure. This will give you all the orphans, which you can now drag-and-drop into the correct positions in your structure.

## Focus on a Specific Level

**Common use cases:**

Often you have a multi-level hierarchy, but are primarily interested in issues on a particular level. For example, you are managing a team that works together with a number of other teams, and you want to see how the scope you are responsible for fits into the full hierarchy. So you would like to see a set of issues you are interested in, and then show the child issues below them and the parent issues above. For example, in a hierarchy where you have Initiative -> Epic -> Story -> Sub-task, you want to see the Stories from a certain sprint and all the related issues - both parents (epics and initiatives these stories belong to) and sub-tasks.

**Performance considerations:**

One of the options would be to build a full hierarchy as described in the [Top-Down Approach](#) and then filter it to show only the part relevant to the user. This approach has one significant drawback - it means you would need to build a really large structure first, and then hide part of it. Even though the issue count in the resulting structure may be low, it will still have a significant impact on performance.

A better approach in this situation would be 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. For example, you can 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 Grouper, 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 want.

## Visualizing the Entire Scope

### Common use cases:

In some situations, you need to see a specific scope of issues arranged into hierarchy. For example, you need to arrange issues in a similar manner as we used above (initiatives > epics > stories > sub-tasks), but only from a particular project, component, version, team, etc.

### Performance considerations:

The approach to building this kind of structure is very similar to the top-down approach we discussed above. If you make sure to use the correct level settings, the performance impact will be comparable, so the implications are the same.

**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 generators to pull in the child issues like we've done before. The important difference in this case is that as you add Extenders, you will be getting duplicate issues - one instance of the child issue is added to the top level by the Inserter and then another is added by the Extender. To hide the duplicates, proceed to the next step.



It's very important to set level parameters for extenders 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 Extenders 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.

To illustrate, let's use the same example (Initiatives, Epics, Story, Sub-Task). If you do not set the levels for extenders, here is what you will get: On the top level you will see the results of the Inserter. On the second level, you will see the results of each Extender - epics under initiatives, stories under epics, sub-tasks under stories - all duplicates. On the third level, you will see stories under epics and sub-tasks under stories - the third time we see them. And on the fourth level you will see the sub-tasks for the fourth time under stories.

Even though all the duplicating items can be hidden (see the next step), we still had to build a really large structure first, which will negatively affect performance. If you do set the levels, we'll only get one set of duplicates for each Extender.

3. Add the Remove Inserter/Extender Duplicates Filter generator - this will hide the issues added to the top level by the Inserter, if they were also added later by an Extender. Any issues that are not duplicated will remain on the top level.
4. The Extenders 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.

## Things to Remember When Using Automation

### Use Levels Settings Where Possible

This can help to clean up your hierarchies. You will see less decision panels (as different generators will work on different levels), and Structure will not need to run unnecessary checks, which are generally fast but can add-up for large structures.

### Keep an Eye on the Number of Issues

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.

### Keep Structure Sizes as Small as Possible

Small structures use less resources, even when you have a lot of them. It's better to create several smaller structures, custom built for specific purposes, teams or people, rather than building one big structure for everything.

## General Performance Considerations

### Automation

Improperly or poorly configured [Automation](#) can have an impact on performance.

- To learn more about how Automation can affect performance, see [Automation Overview and Performance](#).
- To learn more about best practices for using and releasing automation, see [Building Efficient Structures With Automation](#).

### Using Structure Progress, Totals and Formula Columns

Another source of performance load can be the use of Progress column, Totals columns and Formula columns. The Jira fields by themselves used in columns have little impact on performance, as they are loaded only for the structure elements which are currently displayed, plus a number of items below and above (to ensure smoother scrolling).

But for progress, totals and some formulas, the values will be loaded for the entire hierarchy, because such columns aggregate data across the entire hierarchy. For structures with thousands of issues, this can become noticeable; however, it is still much less of an impact when compared to loading the hierarchy itself.



If you have a really large instance and your structures will contain thousands of issues, a good practice is to set up a staging server with a snapshot of data, so users can try building their structures there first. In our experience, 80% of the situations where there were some performance issues happened while the users were experimenting with Automation and built really large and complex structures by mistake.

## Structure on Issue Page

Users can see structures on the main Structure Board, accessible through the top Jira menu; on Jira dashboards; project page and also on the Issue Page. Opening a structure built with the use of automation in any of these locations will require generation of this structure. This is pretty obvious for the main Structure Board or the Dashboard, but less intuitive for the Issue Page.

For this reason, the default structures you select can have a significant impact on overall performance. If a really large structure is selected as [the default structure](#) for a project, whenever a user opens the Issue Page for an issue from this project, this large structure will be loaded. For high-traffic projects, this may affect the performance quite 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](#).

You can also [remove the Structure widget from the Issue Page](#) completely.

## S-JQL

Another non-explicit structure loading will happen if you use [S-JQL](#) - 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 some saved filter with S-JQL for a large structure and use it heavily somewhere, this may start affecting performance too.

## Automation and Synchronizers

One of the most important features of Structure is its ability to visualize large amounts of issues in the hierarchical form based on the relationships which exist between the issues, the field values and other criteria. For example, you can automatically place stories under their epics, and sub-tasks under their parents, or group issues by Sprints, Fix Versions or assignees.

There are two mechanisms for building such structures: [Automation](#) and [Synchronizers](#).

- Automation is the preferred option, which we recommend using whenever possible.
- While there may be some exceptions, Synchronizers are rarely a better option. Synchronizers were the original mechanism for building dynamic structures, but it was replaced by Automation in Structure 3.0 and is gradually being faded out.

To learn more about the differences between Automation and Synchronizers, and the considerations you should take when selecting which to use for your project, see:

## Automation Overview and Performance

Automation allows you define special rules, known as generators, that tell Structure what issues to show you from Jira and where to place them within the structure. Generated can be added right into the structure (unlike Synchronizers which are added via the Manage Structures page).

Once the user opens the structure, the generators are executed one by one, loading and arranging issues. While the structure remains open, they will keep running, monitoring Jira changes and updating the structure accordingly until the structure is closed. There are a couple of key points to keep in mind regarding generators:

- Generators do not run unless the structure is open.
- The issues themselves are not stored in the structures. Instead, they are loaded each time the structure opens, based on the existing generators.
- Generators can only update Jira data as a result of a user's explicit action. A user has to move an issues in a structure to trigger some update (and only if the generator is configured to allow changes via Structure).

## Automation Performance Considerations

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](#), 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.

For more details on the efficient usage of Automation, please refer to our [Building Efficient Structures With Automation](#) guide.



To reduce the risk of Automation affecting the Jira performance in general, we have introduced the Automation Timeout feature, which stops a generation if it exceeds a certain time period. If a user creates a structure that has loading time which exceeds the set threshold, the generation process will be stopped and the user will be able to adjust the generators settings. See [Paused Generation](#) for more details.

## Synchronizers Overview and Performance



### Dark Feature - Synchronizers are hidden by default

We recommend using [Automation](#) and/or [Effectors](#) instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

Starting from version 5.0, Synchronizers are hidden from the UI for all new installations, as well as existing installations that had no synchronizers installed at the time of the upgrade. To remove synchronizers from an existing installation, or to enable synchronizers (not recommended), see [Advanced Configuration and Dark Features](#).

Synchronization is a process for keeping a structure's issue hierarchy in sync with some other issue properties. Synchronizers run continuously in the background, tracking changes both in Jira and in structures, and when there is a change, make the required updates to one or the other. For example:

- When a Filter synchronizer detects a change in a Jira issue, it will check whether any changes should be done to the structure (add a new issue that now matches the filter JQL, or remove one that no longer does).

- When a Link synchronizer detects changes to issue links in Jira, it will check for those issues within the structure and adjust their position accordingly. Additionally, if you change the structure (for example, move a child issue under a new parent), it will remove the existing link and create a new one.

Unlike Automations, Synchronizers are adding and removing issues from the structure. If you switch off or remove the synchronizers, the synchronization will stop and all of the issues will remain in the structure, in their last positions. You can edit it further manually.

## Potential Conflicts

Since synchronizers run as independent processes, each responsible for synchronizing based on a single rule, they can interact with each other. This may result in some unexpected changes, both within the structure and Jira. This is one of the main reasons we chose to replace them with Automation. Here is one common example:

A user wants to see the issues that are the result of a certain JQL query on the top level of the structure, and all issues linked to them on the level below. The user also wants the top level issues removed from the structure if they no longer pass the JQL. So the user sets up 1) a Filter synchronizer with the Remove option to add (and remove) issues based on the JQL and 2) a Links synchronizer to pull in linked issues. Here is what may happen as a result:

1. Filter synchronizer pulls in the top level issues.
2. Link synchronizer, seeing there are some issues added to the structure, checks if they have any issues linked to them with the selected link type and pulls them in.
3. Filter synchronizer sees there are new issues pulled in and checks if they pass the JQL filter. The child issues do not pass the query of the parents, so they are removed.
4. Link synchronizer sees that the issues were removed. It interprets this action to mean those child issues are no longer needed, so it removes the links between the parents and the children.

As a result, the user would lose a number of links, without realizing it. There are ways to avoid such situations, but this will require a higher level of experience and technical knowledge. You can expect some users will make similar mistakes.

It is also possible for synchronizers to reverse each other's changes, creating an infinite loop. For example, if you have several structures with the same sub-sets of issues, synchronizers in one structure may react to synchronizes in another structure. While there are [built-in mechanism to stop this](#), it may still create some unintended consequences before the safeguards are triggered.

Fortunately, there is a way to quickly revert such changes if they occur with the synchronizers Undo feature - but it is still desirable to avoid these situations whenever possible.

## Synchronizers Undo

To revert changes done by synchronizers, do the following:

1. Go to Jira Administration | Structure | Support.
2. Click View Synchronizer Audit Log.
3. Specify the time period when the synchronizer made the unwanted changes.
4. If you know what structure was affected, specify the structure id. You can see the structure id in the URL when you have the structure open.
5. If you know the exact synchronizer which actions you need to undo, specify the id of this synchronizer. You can find it if you go to the Synchronizers Settings page of a particular structure and hover over any of the links in the Operations column. You'll see the URL with the id in the browser status bar.
6. Click the Show button to see the list of all changes done by the synchronizer. The changes are done in batches, so you'll see the number of "actions" in each batch - you can click Expand to see all the details.
7. Select the ones you'd like to undo and switch to the Undo tab.
8. Review the list of actions which will be reverted and click Undo.

## Synchronizers Performance Considerations

When it comes to performance, there are a some things you need to keep in mind:

- Enabled synchronizers run continuously in the background. This means even when a structure is not being used, its synchronizers will continue running and making updates. If you have a large number of structures with synchronizers (a few hundred), their total impact may become noticeable. For this reason, it's important to disable synchronizers for structures which are no longer used. Better yet, delete or archive the structures, since this disables any synchronizers.
- All synchronizers run in a single thread. They make their updates one by one, which means if there is a long queue of updates, there may be quite a long delay between the time you change something in Jira or Structure and the time you see the synchronizers reaction to it. In some extreme cases, it may take several hours and, if the system is slow, the time may grow as more and more updates are added to the queue.

## Mixing Automation and Synchronizers

While it is not recommended, it is possible to use Synchronizers together with Automation. If you choose to do this, keep in mind that Automation sees the results of Synchronizers, but not the other way around:

- If you use a Filter Synchronizer to pull issues into a structure, an Extend generator will see these issues and add any issues linked to them.
- If you use an Insert generator to pull issues into a structure, a Link Synchronizer will not add any child issues to the structure. Synchronizers do not "see" the dynamic content added by Automation.

## Structure Extensions Performance Considerations

Structure.Testy and Structure.Pages should have very limited effect on system performance.

Structure.Gantt, because it works with live Jira data and performs a number of calculations, may affect performance and require some additional attention to minimize its impact.

## Structure.Gantt

### Working With Jira Data

Structure.Gantt uses actual Jira data for visualization of tasks on the timeline. Users can define which custom fields are used for Start and Finish dates, what data is used for task estimation (Jira time tracking info, story points or some other fields) and what Jira link types are used for different types of dependencies.

When a user changes any of these parameters in the Gantt chart (by resizing or moving task bars, adding or removing dependencies, etc.), the user is modifying the corresponding Jira data. While a sandbox mode is planned for the future, currently all the changes are done in real time.

### Structure.Gantt Performance

Some operations and configurations may require additional calculations by Structure.Gantt, which could affect performance. While for smaller structures this shouldn't be a problem, for larger projects, these calculations can add up, so it's recommended to try such configurations on a staging server first.

- The Gantt chart visualizes tasks on the timeline, and with certain settings this may require some calculations. For example, you can use Structure formulas for calculating Start/Finish dates, estimates or progress.
- Visualizing tasks with dependencies may also require certain computational effort.
- Resource allocation is calculated by Structure.Gantt.
- Resource Leveling is a resource-intensive operation. See [Structure.Gantt 2.0 Release Notes](#) for information about testing and customizing this feature for best performance .

For more information please read the [enterprise deployment notes](#).

## 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.

## Reading the Performance Audit Log

The following breaks down how the PAL file is structured.

### Basic Instance Information

Lists the version of Jira and Structure.

### 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](#) 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

### 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](#).
- 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](#).
- 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.

## 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.

## Attribute Service

This section contains some technical information used by ALM Works for further troubleshooting.

## Saved Filters with S-JQL

If you are using [S-JQL](#) 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.

## Gantt Settings

This section lists those structures that have Gantt charts configured and the configuration details.

### 3.17.2 Backup Strategy

- ✔ Prior to version 3.0, Structure used to store data separately from JIRA data and it was not included in the general System Backup. That called for a separate backup strategy. With version 3.0 and later, this matter is simplified.

## General Approach

Structure data is backed up along with JIRA data when you make full system backup.

However, Structure can back up and restore its data separately. This allows you to roll back Structure-related changes without affecting other JIRA data and generally safeguards your structures.

The following backup strategy is sufficient in most cases.

### Option 1. Automatic XML Backup + Export Directory Backup

This strategy involves two processes:

- **Automatic Structure Maintenance** lets you automatically create full hot backups of the Structure data once a day. The backups are stored in the `export` directory under JIRA home.
- Periodic file-level backup of the `export` directory (or the whole JIRA home) to a different storage device increases the safety of the backups. This part should be configured manually by the server administrator.

This is the recommended strategy.

- ✔ When you install Structure, automatic daily backups are enabled by default. You only need to make sure that backup files that will appear in the `export` directory are stored safely.

## Option 2. Manual / API-Triggered XML Backup

You can manually back up structure through [Structure Backup](#) menu.

If automatic Structure maintenance does not suit you and you have resources to develop your own mini-plugin for backup strategy, you can automatically back up Structure data through the [Structure API](#) (use `StructureBackupManager` interface).

## Restoring from XML Backup

See [Restoring Structure from Backup](#) for instructions.

## Incremental and Differential Backups

As Structure database is typically not large, full backup is recommended.

Structure XML backup/restore does not support incremental backup, but you can use your operating system tools for incremental or differential backup of the files in `structure` directory.

### 3.17.3 Gradual Deployment

In an enterprise environment where Jira is already in production and used every day, deploying any new add-on may initially be disruptive to users.

To ease this transition, we recommend a gradual deployment, as described below.

#### Start with Champions First

Structure can be limited to a number of users – see [Restricting User Access to Structure](#). The users who do not have access to Structure don't see Structure's footprint in Jira in any way (with one exception, see below).

A common path to gradual deployment is:

1. Create a group called **structure-users** and restrict access to Structure only to that group.
2. Add to the group those individuals who initially championed getting Structure for your company and anyone else who actively wants to use it.
3. Let them use Structure and spread the word.
4. Once it is decided that everybody wants to use Structure, remove these restriction.
5. Don't forget to advise everyone to check the [Getting Started](#) page.

In the same way, you can gradually enable Structure project-by-project. See [Who Has Access to the Structure](#) for details.

## Turn Optional Functionality Off

Another way to limit the initial deployment of Structure is to [turn off some optional features](#). Here are a couple of examples:

- **Activity Streams.** For technical reasons, even if a user does not have access to Structure, they will still see "Structure" as a possible Activity Streams Provider (although they won't see any events coming out of it). If you are initially limiting the number of Structure users, it may be a good idea to remove Structure data from the activity stream.
- **Automation.** Structure's automation feature can place an additional load on servers, particular as users are first learning to optimize their structures. You can limit the access to Automation by [changing permission to access Automation](#).

## 3.18 ScriptRunner and Structure Cookbook

[ScriptRunner](#) is an app by [Adaptavist](#) 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.

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](#).

### 3.18.1 Use-cases

- [Creating \(multiple\) New Structures Programmatically](#)
- [Creating Generators with ScriptRunner](#) - 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](#) - 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](#) - 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](#) - 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](#) - 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](#) - This script will create a table showing users, each issue they logged work on and how many hours were logged for each particular issue.

### 3.18.2 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
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

@WithPlugin("com.almworks.jira.structure")
```

```
@PluginModule

StructureComponents structureComponents

def plugin = com.atlassian.jira.component.ComponentAccessor.pluginAccessor.get

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 b
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 appl
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
LongArray onlyIssues = new LongArray()

// This variable will hold the rows that that are eventually removed from the
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.JqlQue

// 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 boolea
helper.matchIssues(onlyIssues, query, true, matchingIssues);

// Now we are iterating over our structure again, row by row, to translate the
for (LongIterator ri : forest.getRows()) {
    StructureRow row = rowManager.getRow(ri.value())
    ItemIdentity itemId = row.getItemId()
```

```
        if (CoreIdentities.isIssue(itemId) && matchingIssues.contains(itemId.getLo
            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.si
```

### 3.18.3 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.almworks.jira.structure'
)

def
loader = plugin.getClassLoader()

def
StructureAuth = loader.loadClass(
  'com.almworks.jira.structure.api.auth.StructureAuth'
)

def
JiraUser = loader.loadClass(
  'com.almworks.jira.structure.api.permissions.PermissionSubject$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.g

if
(resync) {

```

```

                                syncManager.resync(sync.instanceId,
true
,
null
)
                                }
else
{
                                syncManager.setAutosyncEnabled(sync.instanceId,
true
)
                                }
                                }
else
{
                                syncManager.updateSynchronizer(sync.instanceId, sync.g
                                }
                                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

```

```

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>"
)

```

### 3.18.4 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

```

```

@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

log.warn "no existing Structure found"
// It is important to note that without the saveChanges() call at the end, the

structureManager.createStructure().setName("testScript").saveChanges()
} else {
log.warn "found a pre-existing structure"
}

```

### 3.18.5 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
jq1 =
  'project = stmb and type = story'

// this is the JQL query you want the generator to execute
```

```
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.structure: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.E
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.

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.SO
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.S
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,
```

```
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

def
permission = PermissionLevel.valueOf(
"ADMIN"
)

// the false flag is optional, it is a variable whether we want to search arch

// we are getting the first (zeroth) element because this method returns a lis

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

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(currentStruc
//resolving forest source for structure

forestSource.apply(
new
ForestAction.Add(generatorItem,
0
,
0
,
0
))
```

### 3.18.6 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
```

```

@WithPlugin('com.almworks.jira.structure')

StructureComponents structureComponents = ComponentAccessor.getOSGiComponentIn

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>

map.collect {'<tr><td>' + Util.htmlEncode(it.key as String) + '</td><td>' + it

'</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.

### 3.18.7 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 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](#) 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

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 + other

    count++
}
remIs
?:null

```



You should replace the value in `baseUrl` with the actual URL of your local instance.

### 3.18.8 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.getPlug  
'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

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><
+
    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>'

```

### 3.18.9 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(),

        searchResult.issues.each{issue -> labelManager.addLabel(user, issue.id

    }
}

```

```
}  
}
```

## 3.19 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**.

### Clear Caches

---

Clearing caches might be needed if a cached value is incorrect

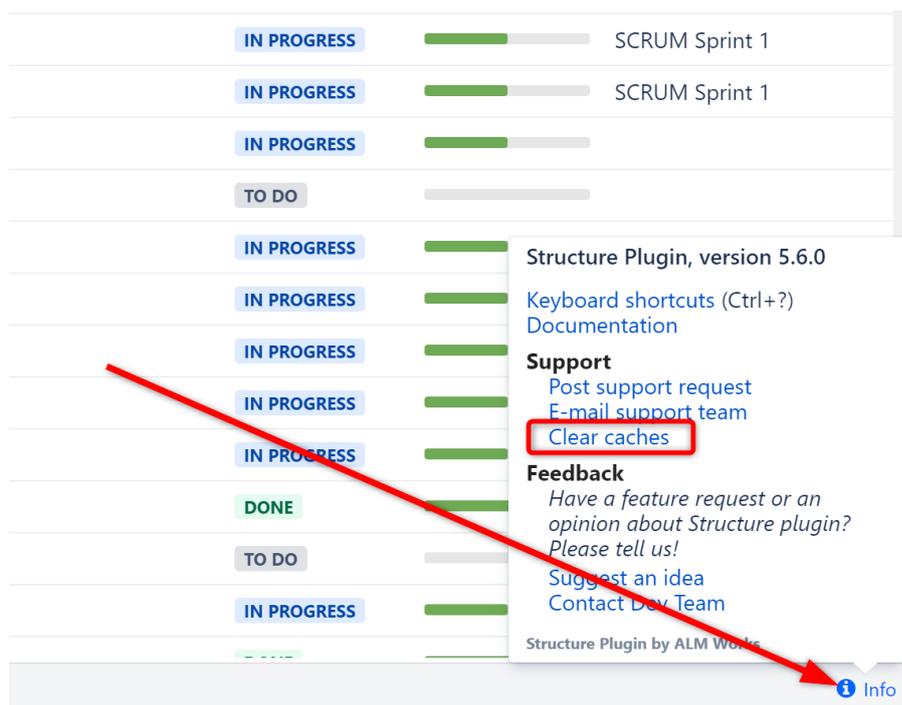
Action

Clear all caches on this node ▼

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.

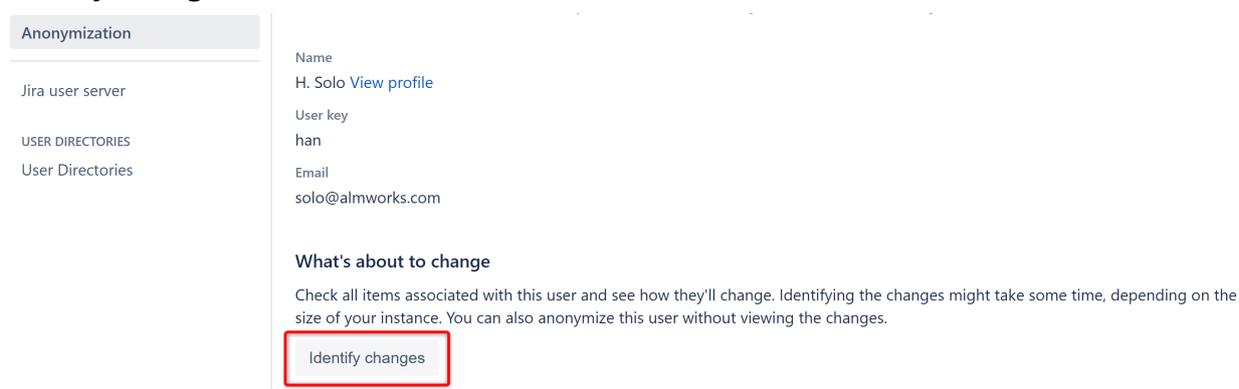


## 3.20 Anonymizing Users

When you [Anonymize a user](#) 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.

### 3.20.1 Review User's Structures

To review the structures and/or views owned by the user, on the Anonymization screen, click **Identify changes**.



Identify changes in Anonymization

Any shared structures or views will appear under the **Transferred items** list. These will be transferred to a new user after the anonymization.

### What's about to change

#### Transferred items

We'll transfer these items to a different user, as they can't work properly on their own.

Item	Number	
Structure "My First Top-Down Automation" owner	1	<a href="#">Structure</a>

Structured to be transferred to another user

Any private structures or views (not visible to anyone but the user) will appear under the **Deleted items** list. These will be deleted.

#### Deleted items

These items are specific to this user, so we'll just delete them.

Item	Number	
Private structure "Text Query Inserter"	1	<a href="#">Structure</a>

Structures to be deleted



To review any of the structures on these lists, click the **Structure** link to the right.

## 3.20.2 Transfer Ownership

Ownership for any shared structures or views will be transferred to the user you select in the **Transferring ownership** section.

#### Transferring ownership

Some items can't work properly on their own, so we'll need to transfer them to a different user.

New owner

[Anonymize](#)

Transfer ownership

## 4 Structure Developer's Guide

### 4.1 Structure Developer Documentation

#### 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](#) and [Java API reference](#).

#### 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](#).

#### Extending Structure

You'd like to extend Structure, adding functionality to the plugin itself – read documentation about [extending Structure functionality with additional plugins](#).

#### 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](#) and [Structure REST API](#).

### 4.2 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.

### 4.2.1 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.

### 4.2.2 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.

## 4.3 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.

### 4.3.1 Setting Up the Integration

To start using Structure in your plugin:

#### Add dependency to your pom.xml

Figure out the [version of the API](#) 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](#)

## 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"/>
```

## 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](#) 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](#). For a standalone plugin, which can work without Structure, read about [Making Structure Dependency Optional](#).

## 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](#) 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](#), which provides specific implementations of these collections.

See [API Usage Samples](#) 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.

## 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](#) 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](#) 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](#) for a more complete example.

**Next:** [Making Structure Dependency Optional](#)

## 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 `resolution:=optional` classifier. ([Add Import-Package to control API compatibility](#) 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 available?

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("co
m.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.structure
(structureId)));
    } catch (StructureException e) {
        return null;
    }
}
}
}

```

### 4.3.2 Structure Services

This page lists public services provided by Structure API. All these services are available from [StructureComponents](#) instance.

#### Services to Start With

Use ...	to ...
<a href="#">StructureManager</a>	Create and delete structures, modify structure properties such as name or permissions. (But not to work with the structure's content.)
<a href="#">ForestService</a>	Access forests for reading or changing.
<a href="#">StructureAttributeService</a>	Retrieve attribute values for given rows in a given forest.
<a href="#">RowManager</a>	Extract item information for rows read from a Forest.
<a href="#">FolderManager</a>	Create folders or change folder properties.
<a href="#">GeneratorManager</a>	Create generators or change generator properties.

#### More Power

Use ...	to ...
<a href="#">StructureConfiguration</a>	Change global Structure add-on configuration.

<a href="#">StructureViewManager</a>	Create and manipulate views.
<a href="#">StructureSyncManager</a>	Manage synchronizers.
<a href="#">StructureBackupManager</a>	Backup complete Structure data to a file or restore it back.
<a href="#">StructureFavoriteManager</a>	Read or change which structures are favorite of which users.
<a href="#">PropertyService</a>	Store arbitrary properties.
<a href="#">StructurePropertyService</a>	Store arbitrary per-structure properties.
<a href="#">AttributeSubscriptionService</a>	Create a subscription for a set of attributes and rows and load data for them asynchronously.

## Extreme Power

Use ...	to ...
<a href="#">ItemTracker</a>	Track recorded changes that happened to items (in JIRA Data Center – on all nodes of the cluster).
<a href="#">ItemResolver</a>	Convert <code>ItemIdentity</code> into an object representing that item.
<a href="#">IssueEventBridge</a>	Listen for or report issue events.
<a href="#">StructureQueryParser</a>	Parse an S-JQL query.
<a href="#">StructureQueryBuilderFactory</a>	Build an S-JQL query via Builder pattern.
<a href="#">ProcessHandleManager</a>	Manage feedback page for asynchronous processes.
<a href="#">SyncAuditLog</a>	Access or manage Synchronization Audit log.
<a href="#">StructureJobManager</a>	Run a job asynchronously.

Use ...	to ...
ScheduledJobManager	Schedule a periodical job to run asynchronously (only on a single node in a cluster).

### 4.3.3 Building Forest Specification

A forest specification, or `ForestSpec`, 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](#) structure, it could be a result of query or some other types of forest source.

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
<i>Base Content</i>	
Structure #123	<code>ForestSpec.structure(123)</code>
Result of a JQL query	<code>ForestSpec.sQuery("jql", "priority = Blocker")</code>
Result of a text query	<code>ForestSpec.sQuery("text", "text to find")</code>
<i>Adjusted Content</i>	
Structure #123, sorted by Priority	<code>ForestSpec.structure(123).transform(CoreStructureGenerators.SORTER_ATTRIBUTE, ImmutableMap.of(</code>

Desired forest	ForestSpec expression
	<pre>                 "attribute", (Object) ImmutableMap.of("id", IssueFieldConstants.PRIORITY, "forma t", "order")                 "desc", true             )         ); </pre>
Structure #123, skeleton only (without dynamic content)	<pre>ForestSpec.skeleton(123)</pre>
Structure #123, with title row	<pre>ForestSpec.skeleton(123).withTitle()</pre>

More details are available in [Javadocs for ForestSpec](#).

#### 4.3.4 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.

#### 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.

## 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](#).

```
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.

## 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.

## 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`.

## 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.

## 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.

### 4.3.5 Changing Structure Content

Updating a structure can be done through the same `ForestSource` interface that was used for [Reading Structure Content](#). In this article, we're assuming that you've got `forestSource` local variable that you've created according to instructions in the previous article.

#### 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.

## 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));
```

## 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.

## 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: YX 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.

## 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.

## 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.

### 4.3.6 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`.

## 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.

## General Approach to Loading Values

Let's assume that, after [Reading Structure Content](#), 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().getForest().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);
```

**i** 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);
    ...
}
```

### 4.3.7 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](#) for general ideas about updating a structure.

## Create the Folder entity

```
long folderId = myStructureComponents.getFolderManager().
createFolder(Folder.named("My Stuff").build());
```

The folder is now stored in the database.

## Define folder's identity

```
ItemIdentity itemId = CoreIdentities.folder(folderId);
```

## Add folder to structure

```
forestSource.apply(new ForestAction.Add(itemId, 0, 0, 0));
```

### 4.3.8 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](#). 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.

#### 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.

#### 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.

## 4.4 Extending Structure Functionality

You can extend Structure add-on's functionality with your own add-on by using one of the available extension points.



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

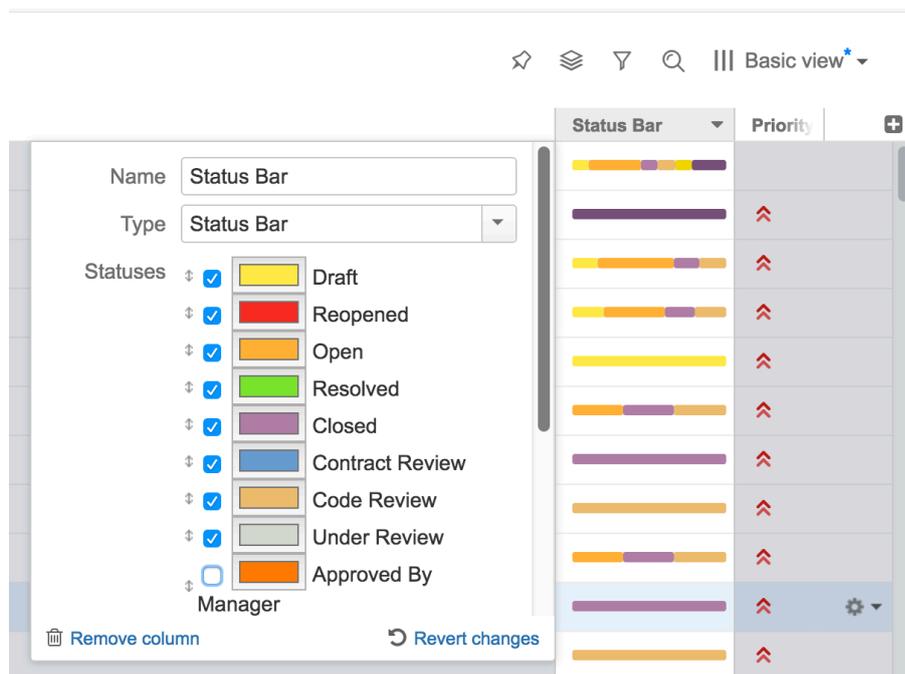
If you're interested in these topics but cannot find documentation or need help, please write to [support@almworks.com](mailto:support@almworks.com) and we'll provide advice.

### 4.4.1 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](#).



## 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](#) 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](#), consulting the other chapters as necessary. For the complete treatment, please continue reading from top to bottom.

## 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.

## 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](#). 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`.

## 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` 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.

**i** 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;
    }
}

```

## 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="attr
ibute-loader:Status Bar Column"
                                class="com.almworks.jira.
structure.sbcolumn.StatusBarAttributeProvider"/>
```

## 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](#) 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` 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
<code>api.ColumnType</code>	<p>The <b>column type</b> is the gateway between Structure and your code. The column type is registered with Structure and has the following responsibilities:</p> <ul style="list-style-type: none"> <li>• creating column presets for the "Add Column" menu;</li> <li>• creating the column preset used when switching to your column type from a different type;</li> <li>• creating <code>Column</code> and <code>ColumnConfigurator</code> instances for given column specifications.</li> </ul>
<code>api.Column</code>	<p>The <b>column</b> 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.</p>
<code>api.ColumnConfigurator</code>	<p>The <b>configurator</b> is responsible for the column configuration panel as a whole. Its most important task is to create <code>ColumnOption</code> instances.</p>

Class or Function	Description
<code>api.ColumnOption</code>	The <b>option</b> 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.
<code>api.registerColumnType(columnType, columnKey)</code>	Registers a column type with the Structure, making it responsible for handling the given column key (see below).
<code>api.registerColumnGroup(parameters)</code>	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": ["#fc3f3e", "#fce94f", "#ef2929", "#8ae234",
"#ad7fa8", "#729fcf"],
    "includeItself": true }}}
```

Key	Description
<code>csid</code>	The <b>CSID</b> ("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. <b>Do not change a column's CSID!</b>
<code>key</code>	The <b>key</b> is a string identifying the column type. Structure uses the key to decide which <code>ColumnType</code> or <code>ExportRendererProvider</code> to use for a particular column. The key is required.
<code>name</code>	The column <b>name</b> 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.
<code>params</code>	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 <code>boolean</code> 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
<code>structure.isPrimaryPanel()</code>	Returns <code>true</code> if the column belongs (or will belong, for presets) to the primary panel of the Structure widget.
<code>structure.isSecondaryPanel()</code>	Returns <code>true</code> if the column belongs (or will belong, for presets) to a <a href="#">secondary panel</a> of the Structure widget.
<code>structure.isStructureBoard()</code>	Returns <code>true</code> if the current widget is on the Structure Board page.

Method	Description
<code>structure. isIssuePage()</code>	Returns <code>true</code> if the current widget is in the Structure section of an issue page.
<code>structure. isGadget()</code>	Returns <code>true</code> if the current widget is embedded in a Structure gadget.
<code>structure. isLocalGadget()</code>	Returns <code>true</code> if the current widget is embedded in a <b>local</b> Structure gadget (i.e. a gadget provided and rendered by the same server).
<code>structure. isRemoteGadget()</code>	Returns <code>true</code> if the current widget is embedded in a <b>remote</b> Structure gadget (i.e. a gadget provided and rendered by different servers).
<code>structure. isGreenHopperTab()</code>	Returns <code>true</code> if the current gadget is in the Structure section of an Agile (GreenHopper) board.
<code>structure. isProjectPage()</code>	Returns <code>true</code> if the current gadget is in the Structure tab of a project page.
<code>jira. getAllIssueFields()</code>	Returns an array of JSON objects representing available JIRA issue fields.
<code>jira. getIssueFieldById (fieldId)</code>	Returns a JSON object representing the JIRA issue field with the given ID, or <code>undefined</code> if there is no such field.
<code>getMetadata(key)</code>	Returns the metadata object associated with the given <code>key</code> . 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 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](#) 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` 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](#) and [column groups](#) 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` 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$.append(
      AJS.template('<div class="checkbox"><label><input type="
checkbox">&nbsp;{label}</label></div>')
      .fill({ label: AJS.I18n.getText("sbcolumn.include-
itself") })
      .toString()).find('input');
    var params = this.spec.params;
    this.checkbox$.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](#) 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.

## ColumnType

The `api.ColumnType` 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](#) 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](#) 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](#).

## 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](#) 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](#) is analogous to `AttributeLoaderProvider`, and [ExportRenderer](#) is a mixture of `AttributeLoader` and the client-side [Column](#).

## 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.

## 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 from 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 "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.StatusBarRendererProvider">
  <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](#), by obtaining the underlying POI objects from `ColumnContext.getObject()` using the keys from `ColumnContextKeys.Excel`.

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.StatusBarExcelProvider">
  <column-key>com.almworks.jira.structure.sbcolumn</column-key>
  <export-format>ms-excel</export-format>
</structure-export-renderer-provider>
```

### 4.4.2 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](#) 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](#). You'll need to use the [Structure JavaScript API](#) and register the web resource with your JavaScript code as a widget extension.

## 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...
<code>structure.widget</code>	Structure Board, Structure Gadget
<code>jira.view.issue</code>	Issue Page, Issue Navigator in details view
<code>gh-rapid</code>	Scrum and Kanban boards from JIRA Software
<code>jira.browse.project</code>	Project page (including Structure tab)
<code>structure.printable</code>	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>
```

### 4.4.3 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.

## Example

```
<structure-item-type key="type-milestone" name="itemtype:Milestone"
  weight="100"
                    class="com.almworks.jira.structure.api.item.
generic.GenericItemType">
  <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 <code>itemType</code> part of <code>ItemIdentity</code> .
@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 <code>GenericItemType</code> .
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 <a href="#">Loading Additional Web Resources For Structure Widget</a> ).
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}".

Element	Required	Description
<code>title</code>	Yes	An I18N key for the item creation panel, e.g. "Milestone".
<code>newItemTitle</code>	Yes	An I18N key for the Structure toolbar's "Add item" drop-down menu, e.g. "New Milestone".

## Programmatic Access to Generic Items

Use `GenericItemService` or `GenericItemManager` 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.

## 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](#), accessed or updated programmatically, and re-inserted into the structure.

## 4.5 Accessing Structure Data Remotely

Structure plugin provides REST API, which is primarily used by the [structure widget](#). 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](#).

## 4.6 Reference

### 4.6.1 Structure Developer Reference

### 4.6.2 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](#) to select the correct API artifact, and you can also download Javadoc JARs there.

## Structure API Versions

### Current Versions

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
17.2.0 <a href="#">Javadocs</a>	Jira 7.13+	6.1.0+	"[17.2,18)"	2020-06-03

17.1.0 <a href="#">Javadocs</a>	Jira 7.13+	6.0.0+	"[17.1,18]"	2020-03-20
17.0.0 <a href="#">Javadocs</a>	Jira 7.13+	6.0.0.eap1+	"[17,18]"	2020-01-30
16.16.0 <a href="#">Javadocs</a>	Jira 7.6+	5.6.0+	"[16.16,17]"	2019-09-18
16.15.0 <a href="#">Javadocs</a>	Jira 7.6+	5.5.0+	"[16.15,17]"	2019-07-08
16.14.0 <a href="#">Javadocs</a>	Jira 7.6+	5.4.0+	"[16.14,17]"	2019-06-03
16.13.0 <a href="#">Javadocs</a>	Jira 7.6+	5.3.0+	"[16.13,17]"	2019-03-22
16.12.0 <a href="#">Javadocs</a>	Jira 7.6+	5.2.0+	"[16.12,17]"	2018-12-25
16.11.0 <a href="#">Javadocs</a>	Jira 7.2+	5.1.0+	"[16.11,17]"	2018-10-25
16.10.0 <a href="#">Javadocs</a>	Jira 7.2+	5.0.0+	"[16.10,17]"	2018-08-16
16.9.0 <a href="#">Javadocs</a>	Jira 7.2+	4.6.0+	"[16.9,17]"	2018-03-28
16.8.0 <a href="#">Javadocs</a>	Jira 7.2+	4.5.0+	"[16.8,17]"	2017-12-26

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
16.7.0 <a href="#">Javadocs</a>	Jira 7.2+	4.4.0+	"[16.7,17]"	2017-11-29
16.6.0 <a href="#">Javadocs</a>	Jira 7.2+	4.3.0+	"[16.6,17]"	2017-10-16
16.5.0 <a href="#">Javadocs</a>	Jira 7.2+	4.2.0+	"[16.5,17]"	2017-08-25
16.4.0 <a href="#">Javadocs</a>	Jira 7.1+	4.1.0+	"[16.4,17]"	2017-06-19
16.3.0 <a href="#">Javadocs</a>	Jira 7.1+	4.0.0+	"[16.3,17]"	2017-04-26
16.2.0 <a href="#">Javadocs</a>	Jira 7.0+	3.6.0+	"[16.2,17]"	2017-04-03
16.1.0 <a href="#">Javadocs</a>	Jira 7.0+	3.5.0+	"[16.1,17]"	2017-01-26
16.0.0 <a href="#">Javadocs</a>	Jira 7.0+	3.4.0+	"[16,17]"	2016-12-07

*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](#).*



[Javadocs for the Latest Version](#) — 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](#).

## 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-17.2.0.jar -
DpomFile=structure-api-17.2.0.pom
```

Name	Version	Published
<a href="#">structure-api-16.16.0-javadoc.jar</a>	1	2019-09-19 18:49
<a href="#">structure-api-16.16.0-sources.jar</a>	1	2019-09-19 18:49
<a href="#">structure-api-16.16.0.jar</a>	1	2019-09-19 18:49
<a href="#">structure-api-16.16.0.pom</a>	1	2019-09-19 18:49
<a href="#">structure-api-17.2.0-javadoc.jar</a>	1	2020-06-03 15:44
<a href="#">structure-api-17.2.0-sources.jar</a>	1	2020-06-03 15:44
<a href="#">structure-api-17.2.0.jar</a>	1	2020-06-03 15:44
<a href="#">structure-api-17.2.0.pom</a>	1	2020-06-03 15:44

### 4.6.3 Structure Plugin Module Types

The following module types are added by the Structure plugin:

- [structure-attribute-loader-provider](#) lets you provide the data for new column types in the Structure widget.
- [structure-export-renderer-provider](#) lets you export new column types to printable HTML and Excel files.
- [structure-item-type](#) lets you define a new type of items, which can be used in structures.
- [Generator Modules](#) 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](#) lets you add effectors to the Automation subsystem.
- [structure-effect-provider](#) lets you define a new effect provider, which can be used by effectors.
- [new-structure-template](#) lets you add templates for new structures.
- [structure-query-constraint](#) allows adding new functions to S-JQL language.
- [structure-synchronizer](#) defines a new synchronizer.

#### [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.
@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 <a href="#">AttributeLoaderProvider</a> .

### 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
	Yes	The module descriptor.

Element	Required?	Description
structure-export-renderer-provider		
@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 renderer provider. Must implement <a href="#">ExportRendererProvider</a> .
column-key	No	The column key that this provider is associated with. You can have multiple <code>column-key</code> 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 <code>printable</code> for the printable HTML format and <code>ms-excel</code> for the Microsoft Excel XLS format. You can have multiple <code>export-format</code> 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.

## 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 <code>itemType</code> part of <a href="#">ItemIdentity</a> .
@name	No	The human-readable name of the plugin module.
@class	Yes	The class that implements the support for the item type. Must implement <a href="#">StructureItemType</a> .

## 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.
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](#)).

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

Generator Type	Icon Classes
Groupier	s-fa s-fa-filter
Sorter	alm alm-sort-asc

## 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 <a href="#">Effector</a> .
label	Yes	The name of the effector as it appears to the user.

Element	Required?	Description
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.

### 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.

### 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 <a href="#">EffectProvider</a> .

### 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](#) 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 <a href="#">StructureQueryConstraint</a> .
fname	Yes	Function name, must be unique throughout the system.

## 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.

Element	Required?	Description
@name	No	The name of the module for JIRA administrators.
@class	Yes	The class that implements the template, must implement <a href="#">NewStructureTemplate</a> .
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.

#### 4.6.4 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 [support@almworks.com](mailto:support@almworks.com) if you'd like to contribute or get preliminary access to that API.

### 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.

## REST Resources

- [Structure Resource](#) is used to create and manage structures (but not the content)
- [Forest Resource](#) is used to retrieve and update forests (a structure's content)
- [Item Resource](#) is used to create and update items (issues, folders and, possibly, items of other types)
- [Value Resource](#) is used to retrieve attribute values for a given forest
- [Attribute Subscription Resource](#) allows subscribing to a particular set of values identified by attributes and rows, and receive updates

## Structure Resource

This page describes resources with which you can [list](#), [create](#), [read](#), [update](#), and [delete](#) structures. Structures contain [general information](#) such as name and permissions, but not the hierarchy itself. Issue hierarchy is accessed through the [Forest Resource](#). This page also documents structure [shape](#) and its [fields](#), and the [error entity](#) that may be returned in case of the REST API user error.

Structure resource belongs to **version 2.0** of the API.

/structure/ GET	list structures
/structure/ POST	create a structure
/structure/{id} GET	read structure
/structure/{id}/update POST	update one or several structure fields
/structure/{id} DELETE	delete structure

Quick navigation:

- [Structure Representations](#)
- [Structure Fields](#)
- [Permission Rules](#)
- [Error Entity](#)

## 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](#)

## 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](#):

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 <a href="#">View</a> access level to the structure, otherwise absent.
editRequiresParentIssuePermission	true if the <a href="#">Require Edit Issue Permission on Parent Issue</a> flag is set on this structure, otherwise absent.
permissions	The list of structure <a href="#">permission rules</a> . Present only if the user has <a href="#">Control</a> 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 <a href="#">owner</a> of the structure. Present only if the user is the owner of this structure or if he has <a href="#">Browse Users</a> permission. A string of the form <code>user:USERNAME</code> , where USERNAME is the JIRA user login name. Example: <code>user:jsmith</code> .

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](#).

[Top](#)

## 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 <code>set</code> , case-insensitive.
subject	Identifies the type of the subject to which the rule applies. Must be one of <code>group</code> , <code>projectRole</code> , <code>user</code> , <code>anyone</code> . See below how to identify the subject.
level	

	<p>Access level to set to the specified subject. Must be equal to one of the names of the <a href="#">Permission Level</a> enum constants, case-insensitive.</p> <p>Please note that Control permission is represented by the <a href="#">ADMIN</a> enumeration constant.</p>
--	---

In addition, there are fields to identify the subject.

group

The rule applies to all users within the JIRA group.

groupId	The name of the JIRA group. Example: <code>jira-developers</code> .
---------	---

*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](#) permission.*

user

The rule applies to the user.

username	Name of the user. Example: <code>jsmith</code> for user John Smith.
----------	---

*REST API user can create such rule only if he has [Browse Users](#) 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 <code>apply</code> , 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](#) access level to the referenced structure.*

[Top](#)

## 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 <a href="#">StructureError</a> 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](#)

## 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=$withOwner&limit=100
```

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.

### **i** Who can access this resource

All users who have [access to the Structure Plugin](#). The returned list contains only structures to which the REST API user has at least [View](#) 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 <a href="#">access level</a> . Must be equal to one of the names of the <a href="#">Permission Level</a> enum constants, case-insensitive. <a href="#">NONE</a> is treated in the same way as <a href="#">VIEW</a> .  Please note that Control permission is represented by the <a href="#">ADMIN</a> enumeration constant.
withPermissions	If <code>true</code> , permission rules will be included in the response. Default is <code>false</code> .
withOwner	If <code>true</code> , owner will be included in the response. Default is <code>false</code> .
archived	If <code>true</code> , the returned list can also contain archived structures. Default is <code>false</code> .
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 <code>application/json</code> , <code>application/xml</code> .
Accept	Should be one of <code>application/json</code> , <code>application/xml</code> .

Response

Success

200 OK	Response entity contains the only field, <code>structures</code> , which contains the list of the structure objects, sorted by name.	<code>application/json</code> , <code>application/xml</code>
-----------	--	---

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>
    </structure>
  </structures>
</structureList>

```

```

    <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 <code>issueId</code> 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 <code>issueId</code> 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 <a href="#">Restore</a> operation may be in progress.	

[Other return codes](#) are possible under the normal rules of HTTP communication.

[Top](#)

## POST /structure

```
POST $baseUrl/rest/structure/2.0/structure
```

[Create](#) an empty structure by POSTing to this resource.

### Who can access this resource

Only logged in users who have [access to the Structure Plugin](#) and a [permission to create structures](#).

#### Request

Request entity should contain the new [structure](#). 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](#).

Please note that this resource accepts only JSON structure representation.

#### HTTP headers:

Content-Type	Must be <code>application/json</code> .
Accept	Should be one of <code>application/json</code> , <code>application/xml</code> .

#### Response

##### Success

201 Created	Response entity contains the created structure with fields, including <code>permissions</code> and <code>owner</code> .	<code>application/json</code> , <code>application/xml</code>
----------------	---	---

#### Example 1: minimal structure

```
POST $baseUrl/rest/structure/2.0/structure
```

Request entity	Response entity
<pre>{   "name": "   Test plan" }</pre>	<pre>{   "id": 104,   "name": "T   est plan",   "descripti   on": "",   "permissio   ns": [],   "owner": "   user:admin" }</pre>

## Example 2: structure with some permissions

```
POST $baseUrl/rest/structure/2.0/structure
```

Request entity	Response entity
<pre>{   "name": "Structure with some permissions",   "editRequiresParentIssuePerm ission": "true",   "permissions": [     {       "rule": "apply",       "structureId": 102     }   ] }</pre>	<pre>{   "id": 105,   "name": "Structure with some permissions",   "description": "",   "editRequiresParentIssuePerm ission": true,   "permissions": [     {       "rule": "apply",       "structureId": 102     }   ],   "owner": "user:admin" }</pre>

## Error

400 Bad Request	<p>Structure data is not well-formed (syntax error) or invalid (semantic error.)</p> <p>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 <i>set</i> rule with level set to an invalid value.</p> <p>Invalid structure example: permissions list contains a rule that fails validation.</p> <p>Response entity contains error. Problems with <i>apply</i> rule usually have structureId to indicate the invalid reference.</p>	application/json, application/xml
403 Forbidden	<p>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.</p>	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 <a href="#">Restore</a> operation may be in progress.	

[Other return codes](#) are possible under the normal rules of HTTP communication.

[Top](#)

## 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](#) 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](#). To access the particular structure, the user has to have at least [View](#) access level.

Request

Path parameter:

id	the ID of the structure
----	-------------------------

Query parameters:

withPermissions	If <code>true</code> , permission rules will be included in the response. Default is <code>false</code> .
withOwner	If <code>true</code> , owner will be included in the response. Default is <code>false</code> .

HTTP headers:

Content-Type	Should be one of application/json, application/xml.
Accept	Should be one of application/json, application/xml.

## Response

## Success

200 OK	<p>Response entity contains the created structure along with all of its fields.</p> <p>Field <code>permissions</code> is included if the REST API user has <a href="#">Control</a> permission on this structure.</p> <p>Field <code>owner</code> is included if the REST API user is either the owner of this structure or has <a href="#">Browse Users permission</a>.</p>	<p>application/json, application/xml</p>
-----------	---	--

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

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 <a href="#">View</a> permission on this structure. Response contains error entity.	application/json, application/xml
404 Not Found	If id is not an integer in $1 \dots 2^{63}-1$ . 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 <a href="#">Restore</a> operation may be in progress.	

[Other return codes](#) are possible under the normal rules of HTTP communication.

[Top](#)

## 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.

### **i** Who can access this resource

Only logged in users who have [access to the Structure Plugin](#) and [Control](#) permission on this structure.

#### Request

Request entity should contain those [structure fields](#) 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](#).

Please note that this resource accepts only JSON structure representation.

#### HTTP headers:

Content-Type	Must be <code>application/json</code> .
Accept	Should be one of <code>application/json</code> , <code>application/xml</code> .

#### Response

##### Success

200 OK	Response entity contains the updated structure with all fields, including <code>permissions</code> and <code>owner</code> .	<code>application/json</code> , <code>application/xml</code>
-----------	---	---

#### Example 1: change description of the Global Structure

```
POST $baseUrl/rest/structure/1.0/structure/1/update
```

Request entity	Response entity
<pre data-bbox="204 376 802 613"> {   "description": "Company-wide structure providing the Big Picture." } </pre>	<pre data-bbox="831 376 1433 1451"> {   "id": 1,   "name": "Global Structure",   "description": "Company-wide structure providing the Big Picture.",   "editRequiresParentIssuePerm ission": 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"     }   ] } </pre>

Example 2: changing permission rules

```
POST $baseUrl/rest/structure/1.0/structure
```

Request entity	Response entity
<pre data-bbox="204 1883 496 1984"> { </pre>	<pre data-bbox="528 1883 1206 1984"> {   "id": 105, </pre>

Request entity	Response entity
<pre> "permissions": [   {     "rule": "set",     "subject": "group",     "groupId": "jira-users",     "level": "edit"   },   {     "rule": "apply",     "structureId": 101   } ] </pre>	<pre> "name": "Structure with some permissions", "description": "", "editRequiresParentIssuePermissi on": true, "permissions": [   {     "rule": "set",     "subject": "group",     "groupId": "jira-users",     "level": "edit"   },   {     "rule": "apply",     "structureId": 101   } ], "owner": "user:admin" </pre>

## Error

400 Bad Request	<p>Structure data is not well-formed (syntax error) or invalid (semantic error.)</p> <p>Not well-formed structure data examples: request JSON is syntactically incorrect; JSON contains unknown field; permissions list contains a <i>set</i> rule with level set to an invalid value.</p> <p>Invalid structure example: permissions list contains a rule that fails validation.</p> <p>Response entity contains error. Responses to problems with an <i>apply</i> rule usually have <code>structureId</code> to indicate the invalid reference.</p>	application/json, application/xml.
403 Forbidden	<p>If REST API user is not logged in, does not have permissions to access Structure Plugin, or does not have <a href="#">Control</a> access level to this structure. Response contains error entity.</p>	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 <a href="#">Restore</a> operation may be in progress.	

[Other return codes](#) are possible under the normal rules of HTTP communication.

[Top](#)

## DELETE /structure/{id}

[Deletes](#) the designated structure.

### Who can access this resource

Only logged in users who have [access to the Structure Plugin](#) and [Control](#) permission on this structure.

Request

Path parameter:

id	the ID of the structure
----	-------------------------

HTTP headers:

Content-Type	Must be <code>application/json</code> .
Accept	Should be absent or equal to one of <code>application/json</code> , <code>application/xml</code> .

Response

Success

200 OK	Contains an object with the only field <code>empty</code> with value <code>true</code> .	<code>application/json</code> , <code>application/xml</code>
-----------	--	---

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

```
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 <a href="#">Control</a> 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 <a href="#">View</a> 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 <a href="#">Restore</a> operation may be in progress.	

[Other return codes](#) are possible under the normal rules of HTTP communication.

[Top](#)

## Forest Resource

Forest Resource is responsible for serving forests and forest updates and receiving the forest actions (change commands) from the client.

## 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	<i>required</i>	The URL-encoded JSON representation of <a href="#">ForestSpec</a> . See also: <a href="#">RestForestSpec</a> .
POST content	<i>required</i>	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={"structureId":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.

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.

## 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](#).

```
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
      "rowId": 123,
      "under": 456,
      "after": 0,
      "before": 124
    },
    {
      "action": "remove",
      "rowId": 442
    }
  ]
}

```

## 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.
item.type	Item type (complete key of the module that provides this item's main functionality.)  Use <code>com.almworks.jira.structure:type-folder</code> for folders and <code>com.almworks.jira.structure:type-issue</code> for issues. See also: <a href="#">CoreItemTypes</a>
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 <a href="#">ForestSpec</a> and <a href="#">RestForestSpec</a> .
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 <a href="#">Forest Resource</a> .
------------------------------	---

## 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:

```
"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
<code>successfulActions</code>	A number of actions successfully performed by the server. In this case, it's either 0 or 1.
<code>itemId</code>	The ID of the newly created item. See <a href="#">ItemIdentity</a> .
<code>oldRowIds / newRowIds</code>	Provides mapping from the temporary row IDs used for uploading the action and the real row IDs obtained after the item was inserted.
<code>forestUpdates</code>	Changes to the forest since the version passed in the request.
<code>itemsUpdate</code>	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).

```
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	<p>The ID of the item.</p> <p>If it is just a number, like in the example, it is an issue ID. Note that it is still a <b>String</b> value that contains issue ID.</p> <p>Instead of a number, it can be a canonical notation of an <a href="#">ItemIdentity</a>. For example, to update a folder, use "com.almworks.jira.structure:type-folder/123" where 123 is the folder ID.</p>
item. values	<p>A map of values to be updated. The keys are the same as when the item is created.</p> <p>For updating a folder, use "summary".</p>
items. version	<p>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.</p>

Parameter (see example above)	Meaning
<code>forest.spec</code> and <code>forest.version</code>	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.

## 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](#).

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](#). 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
<code>requests[i].forestSpec</code>	Forest specification that produces the forest from which the rows are taken.
<code>requests[i].rows</code>	Array of row IDs for which values should be loaded.
<code>requests[i].attributes</code>	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](#) and [CoreAttributeSpecs](#).



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.

```
{
  "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
<code>responses[i].forestSpec</code>	Requested forest spec, from which the rows are taken.
<code>responses[i].rows</code>	A list of row IDs for which the values are provided.
<code>responses[i].data[j].attribute</code>	The attribute specification for which the following values are calculated.
<code>responses[i].data[j].values</code>	Array of values. The value at k-th place corresponds to the row at k-th place in <code>responses[i].rows</code> .



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.

## 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](#).

To learn more about attributes, see [Loading Attribute Values](#).

## 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](#), 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](#).

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 <a href="#">Advanced Configuration and Dark Features</a> .
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 <a href="#">Forest Resource</a>	Defines the forest for which the values will be loaded.

Field	Type	Meaning
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
<code>id</code>	Integer	The unique ID of the subscription.
<code>full</code>	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.
<code>fromVersion</code>	<code>RestVersion</code>	Contains signature and version that were used to request the update.
<code>version</code>	<code>RestVersion</code>	Contains signature and version that should be used to request the next update.
<code>data</code>	Array of Objects	Each object represents values for a particular attribute.
<code>data[i].attribute</code>	<code>AttributeSpec</code>	Describes the attribute for which the values are provided.
<code>data[i].values</code>	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.
<code>data[i].outdated</code>	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.)

Field	Type	Meaning
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
```

<b>Query parameters</b>	valuesUpdate, valuesTimeout	Allow immediately loading some values for the created subscription.
<b>Input data</b>	SubscriptionWindow	Passed as the request body, the window defines the subscription parameters.
<b>Response</b>	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
```

<b>Path parameters</b>	ID	Identifies the subscription by ID.
<b>Query parameters</b>	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 <code>true</code> , then only the already calculated values will be loaded, without requesting the attribute subsystem to reload the values in the window.
<b>Response</b>	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
```

<b>Path parameters</b>	ID	Identifies the subscription by ID.
<b>Query parameters</b>	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.
<b>Input data</b>	SubscriptionWindow	The updated subscription window.
<b>Response</b>	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
```

<b>Path parameters</b>	ID	Identifies the subscription by ID.
<b>Query parameters</b>	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.
<b>Input data</b>	SubscriptionWindow	The object with the parts of the subscription window that need to be updated.
<b>Response</b>	SubscriptionData	Contains all the subscription data and value updates if they were requested.

## Delete Subscription

```
DELETE /attribute/subscription/ID
```

<b>Path parameters</b>	ID	Identifies the subscription by ID.
<b>Response</b>	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.

### 4.6.5 Structure JavaScript API Reference

Structure's JavaScript API provides ways to extend the client-side functionality of the Structure plugin.

## JavaScript API Functions

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](#).

#### Parameters

<code>className</code>	<i>string</i>	Class name as string (optional, used for friendly instance names in debugger)
<code>superclass</code>	<i>Object</i>	Superclass reference
<code>prototype</code>	<i>Object</i>	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](#) specification, along with the type-specific parameters and column name.

#### Parameters

type	<i>Object</i>	A <code>ColumnType</code> instance, implementing a specific column type – see <a href="#">ColumnType Class</a>
key	<i>string</i>	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	<i>string</i>	Group key. The same key should be returned by all ColumnConfigurator's <code>getGroupKey()</code> method for all column types which need to appear in this particular group.
options. title	<i>string</i>	Group title
options. order	<i>number</i>	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	<i>string</i>	Item type for which details are registered
ProviderClass	<i>Object</i>	Subclass of <a href="#">ItemDetailsProvider</a> 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);
```

## 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\(\)](#) method.

### 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](#) 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](#) 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

<code>renderingParameters.getAttributeValue()</code>	returns current row's attribute value
<code>renderingParameters.getRowId()</code>	returns current row's id
<code>renderingParameters.getItemId()</code>	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

<code>renderingParameters.getAttributeValue()</code>	returns current row's attribute value
<code>renderingParameters.getRowId()</code>	returns current row's id
<code>renderingParameters.getItemId()</code>	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](#).

## Parameters

<code>attributeSet.requireAttribute (attributeSpec,forestSpec)</code>	<p>Method for collecting required attributes.</p> <p>Parameters are:</p> <ul style="list-style-type: none"> <li>• <code>attributeSpec</code> is the attribute specification object</li> <li>• <code>forestSpec</code> is the forest specification for the forest, from which attribute should be loaded (<b>optional</b>)</li> </ul>
---	--

## About Attribute Specs

`AttributeSpec` defines the attribute and format to be loaded. See [Loading Attribute Values](#) 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.

## 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
<code>{id: &lt;jira-field-id&gt;, format: 'html'}</code>		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.
<code>{id: 'customfield', format: 'html', params: { fieldId: &lt;field-numeric-id&gt; }}</code>		HTML representation of a custom field value.
<code>{id: 'project', format: 'id'}</code>		Project ID for the issues. The <code>id</code> format means either a string or a number, depending on what is being used for identifying the object.
<code>{id: 'editable', format: 'boolean'}</code>		Boolean value telling whether the item can be edited by the user.



See also [CoreAttributeSpecs](#) 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()

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](#) 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 encapsulates 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\(\)](#) 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\(\)](#) for reference.

`getMetadataRequests()`

Returns a JavaScript object specifying the metadata needed by this configurator to set up the UI. See [Requesting and Using Metadata](#) 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](#)

### 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\(\)](#) 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 || '4
2') : '');
    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$)` 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` 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](#) 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](#) 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](#) 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](#) 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\(\)](#) function, define necessary properties and then register provider subclass for the specific item type using [registerItemDetailsProvider\(\)](#).

#### 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](#) 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\(\)](#). 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\(\)](#) on Structure update.

Default `false` value means that details will be refreshed only after successful item editing on Structure panel.

## Methods

init()

Initializer that is called during subclass instance creation.

extendFocusedRowData(rowData)

Returns a [jQuery.Deferred](#) 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`. 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	<i>Object</i>	Holds current row data: <ul style="list-style-type: none"> <li>• <code>'rowId'</code>: <i>Number</i></li> </ul>
---------	---------------	---

- |  |  |   |
|--|--|---|
|  |  | <ul style="list-style-type: none"> <li>'itemId': <i>String</i></li> </ul> |
|--|--|---|

showDetails(rowData, life)

Must be implemented in subclass. The main method - displays the details for the current row in the viewport.

Returns [jQuery.Deferred](#) 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	<i>Object</i>	Row data as returned by <a href="#">extendFocusedRowData()</a>
life	<i>Object</i>	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 <code>life.addDetach(function)</code>

beforeHide(rowData)

Called just before the panel is hidden to perform necessary cleanup.

### Parameters

rowData	<i>Object</i>	Row data as returned by <a href="#">extendFocusedRowData()</a>
---------	---------------	--

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	<i>Object</i>	Row data as returned by <a href="#">extendFocusedRowData()</a>
---------	---------------	--

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	<i>Object</i>	Row data as returned by <a href="#">extendFocusedRowData()</a>
---------	---------------	--

applyWidth(width)

Updates the displayed details content so it fits the given width. Called on changing details panel width.

### Parameters

width	<i>Number</i>	New panel width
-------	---------------	-----------------

onDetailsFocusing()

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](#) is set to `true`.

Returns a [jQuery.Deferred](#) that must resolve when item details has been refreshed.

### Parameters

rowData	<i>Object</i>	Row data as returned by <a href="#">extendFocusedRowData()</a>
life	<i>Object</i>	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 <code>life.addDetach(function)</code>

setDebug(debug)

Controls additional debug information in logs.

### Parameters

debug	<i>Boolean</i>	Enable/disable debug logging
-------	----------------	------------------------------

## JavaScript API Objects

This page lists objects exposed by the Structure API.

### 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](#) subclasses via 'itemDetailsBridge' property.

### Properties

panel\$

jQuery object that contains item details panel element. All item details [viewports](#) are appended to this container.

### Methods

getCurrentRowData()

Returns `rowData` as returned by corresponding [ItemDetailsProvider.extendFocusedRowData\(\)](#) 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	<i>Function (Object)</i>	function that accepts <code>rowData</code> as returned by <a href="#">ItemDetailsProvider.extendFocusedRowData()</a>
----------	--------------------------	--

toggleEnabled(enabled)

Enable/disable details panel layout.

### Parameters

enabled	<i>Boolean</i>	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

<code>listener</code>	<i>Function(Boolean)</i>	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

<code>listener</code>	<i>Function</i>	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

<code>listener</code>	<i>Function</i>	function that performs cleanup
-----------------------	-----------------	--------------------------------

`notifyDetailsEditStarted()`

Should be called when user starts item details content editing. This prevents details refreshing after possible structure updates.

`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	<i>Boolean</i>	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	<i>Object</i>	<a href="#">jQuery.Deferred</a> 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	<i>Boolean</i>	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	<i>String</i>	identifier of entity that stole focus from details panel
focused	<i>Boolean</i>	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.

**Parameters**

stealerId	<i>String</i>	identifier of entity that released focus
preventFocusing	<i>Boolean</i>	indicates that details panel should not be focused even if all focus stealers were released

**4.6.6 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	<a href="#">Structure Board</a> , <a href="#">Structure Gadget</a> See <a href="#">Loading Additional Web Resources For Structure Widget</a> for the recommended way of extending the widget.
structure.printable	<a href="#">Printable page</a>

For details about how to use web resource contexts, see [Atlassian Developer Documentation](#).

**4.7 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.

**4.7.1 Download**

Name	Version	Published
<a href="#">custom-itemtype-2.0.0.jar</a>	1	2020-06-03 15:44
<a href="#">labels-extender-1.0.3.jar</a>	1	2020-06-03 15:44
<a href="#">status-bar-column-3.0.0.jar</a>	1	2020-06-03 15:44
<a href="#">structure-api-examples-6.0.0.zip</a>	1	2020-06-03 15:44

Name	Version	Published
<a href="#">user-item-details-1.0.1.jar</a>	1	2020-06-03 15:44



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.

### 4.7.2 Example List

Sample Plugin	Description
simple-plugin	Very basic demo of using <code>StructureManager</code> .
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.

## 4.8 Archived Documentation

### 4.8.1 Structure 3 API Changes

#### 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 [support@almworks.com](mailto:support@almworks.com)

#### 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:

- 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	<code>com.almworks.jira.structure:type-issue</code>	Issue ID (long)	Default when item type is not specified
Folder	<code>com.almworks.jira.structure:type-folder</code>	Either folder ID (long) or folder i18n name (string)	Folders are introduced in Structure 3
User	<code>com.almworks.jira.structure:type-user</code>	User key (string)	
Generator	<code>com.almworks.jira.structure:type-generator</code>	Generator ID (long)	A generator is an automation rule embedded in the structure.
Page			Confluence pages are added as a type by Structure.Pages extension

Item Type	Module Key	Meaning of Item ID	Comments
	<code>com.almworks.structure.pages:type-confluence-page</code>	Page ID (ID modulo 1e9)	

## 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
A value in the Structure grid is displayed by ...	Column	Column
A column requests from the server ...	Fields	Attributes

## 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>
```

## 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	<pre> AttributeSpecBuilder   .create("customfield", ValueFormat.NUMBER)   .params()     .set("fieldId", 10000) // 10000 - the id of "Story Points" custom field   .build() </pre>
Story Points	<pre> AttributeSpecBuilder   .create("sum", ValueFormat.NUMBER)   .params()     .setAttribute(storyPoints) // storyPoints = the attribute spec for Story Points     .set("distinct", true) // exclude duplicates   .build() </pre>

## 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	<pre> forestSource.   apply(new UpdatableForest Source.Update. Add( </pre>

Operation	Code
	<pre data-bbox="1090 331 1430 539">CoreIdentities. issue(10200), 1 040, 1900, 2000 ))</pre>
Remove rows with IDs 10100 and 10102	<pre data-bbox="1090 622 1430 965">forestSource. apply(new UpdatableForest Source.Update. Remove(     LongArray. create(10100, 1 0102)))</pre>
Move row with ID 1010 as the first row under parent row with ID 1040, before a row with ID 1060	<pre data-bbox="1090 1052 1430 1395">forestSource. apply(new UpdatableForest Source.Update. Move(     LongArray. create(1010), 1 040, 0, 1060))</pre>

## 4.8.2 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. Feel free to reach out to [support@almworks.com](mailto:support@almworks.com) if you need help.

Structure comes with a number of bundled [synchronizers](#), but you can add another synchronizer to the system, allowing Structure users to install it on structures and run export / import.

## Implement StructureSynchronizer

Create your implementation of [StructureSynchronizer](#) interface. Use [AbstractSynchronizer](#) as the base class.

## Define structure-synchronizer Module

Add [structure-synchronizer](#) module to your `atlassian-plugin.xml`, referring to your implementation of the `StructureSynchronizer`.

## Test Thoroughly

Test how your synchronizer works when other synchronizers are also installed onto the same structure.

## 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](#) page.

### 4.8.3 structure-synchronizer

Synchronizer module allows you to plug additional synchronizers into Structure.

#### 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
<code>structure-synchronizer</code>	Yes	The module descriptor.
<code>@key</code>	Yes	Unique module key within the plugin.
<code>@order</code>	Yes	Order of the synchronizer among other synchronizers, whenever a list of synchronizers is present.
<code>@class</code>	Yes	The class that implements the synchronizer. Must implement <a href="#">StructureSynchronizer</a> . It is recommended to extend <a href="#">AbstractSynchronizer</a> .
<code>label</code>	Yes	The name of the synchronizer.
<code>description</code>	No	Description of the synchronizer.
<code>rules</code>	No	The text that is shown at the top of the synchronizer configuration page. Could be a large text.
<code>resource</code> [ <code>@name="form"</code> ]	Yes	A velocity template that contains the form for the synchronizer parameters.

## 5 Structure FAQ

### 5.1 Frequently Asked Questions

### 5.2 Data Center Approved Apps FAQ

Atlassian recently established a new class of marketplace apps built for Data Center. These Data Center approved apps must adhere to new development requirements and undergo additional testing to ensure they meet the unique requirements of large-scale 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.

#### 5.2.1 What are Data Center approved apps?

Atlassian has established new 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 new 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.

To learn more about the importance of using Data Center approved apps, see [the Atlassian Data Center FAQ](#) .

#### 5.2.2 Didn't Structure already work with Data Center?

Structure for Jira, Structure.Gantt, Structure.Pages and Structure.Testy already worked well on Data Center; however, all of these apps have now gone through Atlassian's new approval process.

Receiving Data Center approval means we had to put the entire Structure family of apps through an architectural review as well as a rigorous series of additional tests to prove they can handle the unique requirements of large-scale Data Center environments.

### 5.2.3 Why has Structure made this change?

The entire Structure family of apps already worked well on Data Center, but the Data Center approved apps program is about more than just working well. It's about developing and testing apps to the specific needs of a data center environment – which is very different from the needs of a server environment.

The approved apps program means our apps are held to a higher standard – and as Data Center requirements change, so too will those standards. For us, this means we can continue to provide the best possible experience for all of our customers.

### 5.2.4 What are the criteria for being Data Center approved?

In order to be listed as a Data Center approved app, developers must adhere to new, more-rigorous development and testing criteria, designed to ensure such apps will perform to the unique requirements of large-scale Data Center environments. These additional 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](#).

### 5.2.5 Why should I upgrade to the Data Center approved app version?

Your large-scale Data Center instance has different needs than a traditional server, and you should be using apps specifically tested to function efficiently and reliably in such an environment.

Even for vendors who have developed apps for Data Center in the past, such as ALM Works, the new Data Center approved app program holds us to a higher standard, putting our apps through additional and ongoing testing to ensure we continue to meet the ever-more-demanding needs of Data Center environments.

### 5.2.6 Why is the price different for Data Center approved apps?

Data Center approved apps follow an annual subscription model, based on your Data Center user tier.

To determine the exact pricing for your team, [see the Structure pricing for Data Center](#) and [Structure.Pages pricing for Data Center](#) on the Atlassian Marketplace. [Structure.Testy](#) remains a free extension to Structure on both Server and Data Center platforms. [Structure.Gantt](#) is currently free, but will become a paid app starting with version 2.0. You can find more details [here](#).

### 5.2.7 Do I have to switch to the Data Center approved version? Can I continue to use the server app in Data Center?

You can continue to use your current ALM Works server app until your maintenance period expires. After that time, you will need to upgrade to the Data Center approved app in order to continue receiving support.

Any ALM Works server licenses that are purchased after September 3, 2019 will not work on Data Center.

## 5.3 Cannot Create an Issue With +Next Issue (+Sub-Issue) Because of the Required Fields

### 5.3.1 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.

### 5.3.2 Answer

There are two ways to create new issues when additional fields are required:

#### 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](#) for more information.

## 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](#) 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.

## 5.4 Add-on Manager Says Structure Is Unlicensed

### 5.4.1 Question

I have a valid license installed. Why do I see Structure as **Unlicensed** or having **Action Required** in the Add-on Manager?

### 5.4.2 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](#).

## 5.5 No Check Mark Displayed for a Resolved Issue

### 5.5.1 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?

### 5.5.2 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](#), [Filtering](#), [Progress Column](#)

### 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.

## 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".

## 5.6 Structure Won't Start

### 5.6.1 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?

### 5.6.2 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.

## 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.

See also: [Structure Files Location](#)

## 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.UserCanCreateStructureCondition' 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 [support@almworks.com](mailto:support@almworks.com).

### 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.

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](#) page.

### Incorrect Jira setup

Check the [Jira application logs](#) 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](#). 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](#). 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 [ALM Works support](#).

## 5.7 After an Issue is Moved to Another Project, It Cannot Be Found in the Structure

### 5.7.1 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?

### 5.7.2 Answer

There may be a couple of reasons for this:

#### The New Project Is Not Structure Enabled

Make sure the project where the issue was moved to is [enabled for Structure](#). 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](#) or move the issue to an already Structure-enabled project (such as the original project).

#### The Issue Was Removed Based on Automation Rules

If the structure was built using [Automation](#), make sure the issue's new location satisfies the rules of your [Insert](#) and/or [Extend](#) generators. If it does not, you will need to change your generator rules or move the issue.

## 5.8 User Cannot Access Structure

### 5.8.1 Question

The user cannot see the Structure menu or access any structures. How do I resolve this problem?

### 5.8.2 Answer

To access Structure:

1. Structure has to be [enabled for the user](#) - 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](#) 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.

### 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](http://en.wikipedia.org/wiki/Wikipedia:Bypass_your_cache).

## 5.9 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**.

## 5.10 Using Subtasks and Structure

### 5.10.1 Question

Should I disable sub-tasks to use Structure?

### 5.10.2 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](#) to automatically add Jira sub-tasks beneath their parent issues.

## 5.11 Difference from Sub-tasks

### 5.11.1 Question

How is issue hierarchy provided by Structure different from the standard sub-tasks?

### 5.11.2 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](#).

## 5.12 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](#) when generation times take too long, but you can also limit the access to Automation by [changing permission to access Automation](#).
- If necessary, you can [switch off some parts of Structure](#) to reduce the load (for example, the Structure panel on the Issue Page) and [limit the group of users](#) Structure is exposed to.
- If you experience other performance issues, please contact [ALM Works Support](#).

## 5.13 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. Open the structure [History](#) panel.
2. In the history, find and select the moment when the structure was in the desired state (before the unwanted changes took place).
3. Press CTRL+A (+A on Macs) to select all issues and press CTRL+C (+C) to cut them to clipboard.
4. 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](#) (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.

## 5.14 Can a Structure be Exported to Microsoft Word?

Exporting to Microsoft Word is available through our [integration with Xporter](#).

If you do not have Xporter installed, the following built-in export options are available:

- [Export to Excel](#) - Export your hierarchies into a fully-functional MS Excel spreadsheet.
- [Export to Printable Page](#) - 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.

## 5.15 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.

### 5.15.1 How to Install

1. Download the attached Macro: [FormatTimetrackingData.xlam](#).
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**.

### 5.15.2 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 Structure Troubleshooting

### 6.1 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.

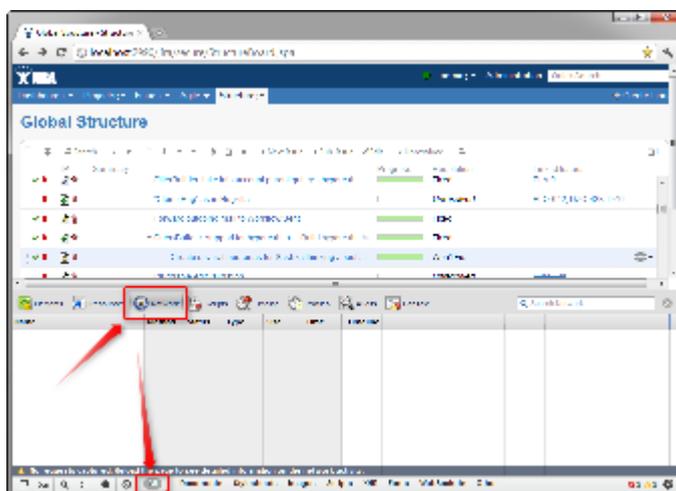
## 6.2 HAR Network Report

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.

**i** HAR stands for [HTTP Archive Format](#), 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](#) 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.)

### 6.2.1 Collecting HAR Report with Google Chrome

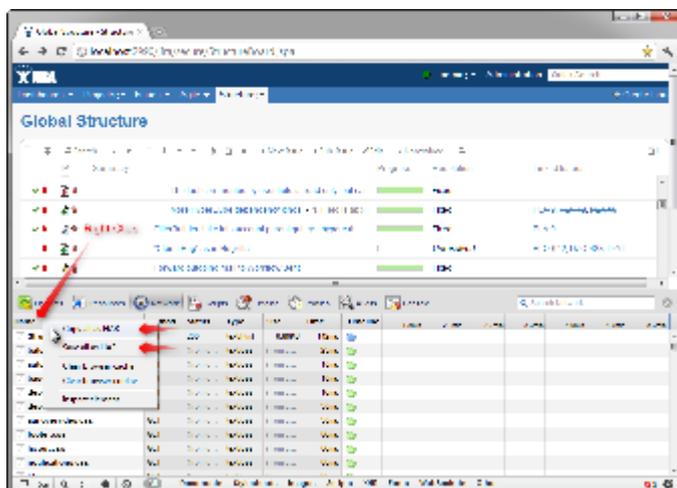
1. Open a new Chrome window and navigate to the page where the problem happens.
2. Press **Ctrl+Shift+I** or use menu **Wrench | Tools | Developer Tools** to open a section with developer tools. Switch to the **Network** tab there. Make sure **All** tab is selected below.



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.

## 6.3 Troubleshooting Synchronizers

**Structure synchronizers** 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.

### 6.3.1 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.

### 6.3.2 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** (until the next JIRA restart) or **permanently**.

To see the list of possible log levels and other general information regarding logging in JIRA, please refer to [JIRA logging documentation](#).

## 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](#) 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](#).

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:

```
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
```

## 6.4 Structured JQL Troubleshooting

If a [Structured JQL](#) 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. Feel free to ask a question on [Atlassian Answers](#) or write to [support@almworks.com](mailto:support@almworks.com) if you need help with S-JQL.
2. Probably, JIRA indexes that are used for searching have become corrupt. Please try to do a [full reindex of JIRA](#) — 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](#).

On this page, you will be able to run a Structured JQL query and collect extensive logs which we in ALM Works can inspect in order to track down the issue.

## 6.5 Collecting Performance Snapshots

Performance snapshots allow ALM Works support team to analyze performance-related problems on your JIRA server without direct access to it.

### 6.5.1 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).

Name	Version	Published
<a href="#">atlas-yourkit-0.2.jar</a>	1	2017-04-05 02:22

### 6.5.2 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](#) for details.

### 6.5.3 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.

### 6.5.4 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.

### 6.5.5 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](#).

### 6.5.6 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.)

### 6.5.7 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](#). 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.

## Download Profiling Agent

Download the ZIP with profiling agent from here: [jira-profiler-v1-yjp956.zip](#) 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](#).

## 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):

```
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](#).

## Attach Profiler Agent to JIRA without Restarting

**i** If you have restarted JIRA with profiling, skip this step.

**i** 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-redis.jar`, located in `<jira_install>/profiler`. The common format for running this program is:

```
java -jar yjp-controller-api-redis.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-redis.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-redis.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.

## 6.6 Sending Files to Support Team

When you need to send files to ALM Works support team, please use one of the following methods (listed in the order of preference).

### 6.6.1 Attach to the Support Request in ALM Works Service Desk (Preferred)

**File size limit: 20 MB**

If the files pertain to a Support Request on <https://support.almworks.com>, please use Service Desk to upload and attach the files to the ticket. Size limit is 20 MB per upload.

### 6.6.2 Send Files by E-mail

**File size limit: 20 MB**

You can send the files to [support@almworks.com](mailto:support@almworks.com). Maximum total attachments size is 20 MB.

If you don't have a preceding e-mail communication with support about the problem in question, please add a short comment or a reference to the problem being diagnosed.

### 6.6.3 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. 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 ALM Works support.

## 6.7 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.

### 6.7.1 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](#) for instructions of manually capturing a thread dump on the server.
- 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.

### 6.7.2 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.

### 6.7.3 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.

### 6.7.4 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.

### 6.7.5 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.

### 6.7.6 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.