Structure for Jira

Structure for Jira 6.4

Exported on 02/01/2021

Table of Contents

1 Structure – Project Management at Scale 47
2 Getting Started with Structure 49
2.1 Before You Begin 49
2.2 Creating Your First Structure 49
2.2.1 Your First Structure 49
2.2.2 Next Steps 50
2.2.3 Building a Structure with Automation 50
2.2.3.1 A (Very) Brief Overview of Automation 50
2.2.3.2 Top Down Automation to Manage Issues Across Multiple Projects 51
2.2.3.3 Bottom Up Automation for Backlog Grooming 56
2.2.4 Building a Structure Manually 60
2.2.4.1 Step 1: Open the Secondary Panel 60
2.2.4.2 Step 2: Search for Issues 61
2.2.4.3 Step 3: Add Issues to Your Structure 61
2.2.4.4 Step 4: Organize Your Issues 62
2.2.4.5 Next Steps 63
2.3 Working with Structure 63
2.3.1 Moving, Adding and Removing Items from a Structure 63
2.3.1.1 Moving Items 63
2.3.1.2 Adding Items 64
2.3.1.3 Deleting Items 64
2.3.2 Adding Columns 65
2.3.2.1 Special Columns and Views 65
2.3.3 The Structure Board 65
2.3.4 Structure in Other Locations 66
2.3.5 Next Step 66
2.3.6 Working with Issues in Structure 66
2.3.6.1 Issue Details Panel 66
2.3.6.2 Editing Issues 67
2.3.6.3 Next Steps 68
2.3.7 Search Filter and Sort 69
2.3.7.1 Search 69

2.3.7.2 Filter ... 70
2.3.7.3 Transformations ... 70
2.3.7.4 Quick Transformations ... 71
2.3.7.5 Next Steps ... 71
2.4 Help and Support ... 71
2.4.1 Available Resources ... 71
2.4.2 Resources Within Structure ... 72
3 Structure User's Guide ... 73
3.1 Navigating Structure ... 73
3.1.1 Basic Concepts ... 73
3.1.2 Structure Menu ... 74
3.1.3 Main Structure Toolbar ... 76
3.1.3.1 Available Actions ... 77
3.1.4 Structure Widget Overview ... 78
3.1.4.1 Navigating Between Items ... 79
3.1.4.2 Switching Between Structures ... 80
3.1.4.3 Using Structure Widget for Searching ... 81
3.1.5 Jira Pages with Structure ... 83
3.1.5.1 Open on Structure Board ... 83
3.1.5.2 Structure Board ... 83
3.1.5.3 Structure on the Issue Page ... 85
3.1.5.4 Structure on the Project Page ... 90
3.1.5.5 Structure on Agile Boards ... 91
3.1.6 Sharing a Perspective ... 92
3.2 Basic Operations ... 93
3.2.1 Adding Items to a Structure ... 94
3.2.1.1 Adding Issues ... 94
3.2.1.2 Folders ... 95
3.2.1.3 Memo ... 97
3.2.2 Moving Items within Structure ... 99
3.2.2.1 Basic Moves ... 99
3.2.2.2 Advanced Moves ... 100
3.2.2.3 Moving Multiple Items ... 100
3.2.2.4 Back-to-Back Moves ... 101

3.2.2.5 Moving Items with Automation ... 101
3.2.2.6 Selecting Multiple Items ... 101
3.2.2.7 Using Cut, Copy and Paste ... 103
3.2.2.8 Using Drag and Drop ... 105
3.2.3 Removing Items from Structure ... 107
3.2.4 Undoing Changes ... 107
3.2.5 Working with Issues ... 108
3.2.5.1 Viewing Issue Details ... 108
3.2.5.2 Creating New Issues ... 110
3.2.5.3 Editing Issues ... 114
3.2.5.4 Bulk Change ... 120
3.2.5.5 Cloning Multiple Issues ... 121
3.2.5.6 Using Jira Actions ... 121
3.3 Automation ... 122
3.3.1 How Automation Works ... 123
3.3.2 Types of Generators ... 123
3.3.3 Generator Scope ... 123
3.3.4 Adding a Generator ... 123
3.3.4.1 Automation Editing Mode ... 123
3.3.4.2 Adding a Generator ... 124
3.3.5 Types of Generators ... 124
3.3.5.1 Insert Generators ... 125
3.3.5.2 Filter Generators ... 130
3.3.5.3 Sort Generators ... 139
3.3.5.4 Group Generators ... 141
3.3.5.5 Extend Generators ... 146
3.3.6 Effectors ... 154
3.3.6.1 Attribute to Issue Field Effector ... 154
3.3.6.2 Status Rollup Effector ... 157
3.3.6.3 Running an Effector ... 161
3.3.6.4 Revert Effector Changes ... 165
3.3.6.5 Managing Effectors ... 167
3.3.7 Generator Scope ... 169
3.3.7.1 Manual Levels Range ... 170
3.3.8 Editing a Generator ... 170

3.3.8.1 Change Scope ... 171
3.3.8.2 Change Settings ... 171
3.3.9 Deleting a Generator ... 171
3.3.10 Paused Generation ... 172
3.3.10.1 Resuming Generation ... 172
3.3.10.2 Changing the Allowed Generation Time for a Structure ... 173
3.3.10.3 Changing the Default Generation Time Limit ... 175
3.3.10.4 System-wide Hard Limit ... 175
3.3.11 Manual Adjustments ... 175
3.3.11.1 Enabling Manual Adjustments ... 176
3.3.11.2 Manual Adjustments are NOT Reflected in Jira ... 177
3.3.11.3 Special Considerations When Using Manual Adjustments ... 177
3.3.11.4 Why is Manual Adjustment Necessary? ... 178
3.3.11.5 Order of Operations with Manual Adjustments ... 178
3.3.11.6 Undoing Manual Adjustments ... 179
3.3.12 Order of Operation for Generators ... 179
3.3.12.1 Generator Types ... 179
3.3.12.2 Generators of the Same Type ... 179
3.4 Search, Filter and Transformation ... 180
3.4.1 Search ... 180
3.4.1.1 Search Modes ... 181
3.4.2 Filter ... 183
3.4.2.1 Filter Is a Transformation ... 183
3.4.3 Transformations ... 184
3.4.3.1 Transformations vs. Generators ... 184
3.4.3.2 Available Transformations ... 185
3.4.3.3 Working with Transformations ... 185
3.4.3.4 Using Transformations ... 185
3.4.3.5 Quick Transformations ... 188
3.4.3.6 Default Quick Transformations ... 192
3.4.4 Pinned Item Mode ... 192
3.4.4.1 What is Displayed in Pinned Item Mode? ... 193
3.4.4.2 Turning Pinned Item Mode On and Off ... 193
3.4.4.3 Limitations Imposed by Pinned Item Mode ... 194
3.4.5 Identifying Duplicate Items ... 194

3.4.5.1 Finding Duplicates ... 194
3.4.5.2 Mark Duplicate Rows ... 195
3.4.5.3 Filtering by Duplicates ... 196
3.4.5.4 Pinning Duplicates ... 196
3.5 Formulas ... 197
3.5.1 Formula Columns ... 197
3.5.1.1 Add a Formula Column ... 198
3.5.1.2 Formula ... 198
3.5.1.3 Variables ... 201
3.5.1.4 Options (Aggregation) ... 201
3.5.1.5 Format ... 202
3.5.1.6 Additional Information ... 203
3.5.1.7 Columns as Variables ... 203
3.5.1.8 Formulas as Variables ... 205
3.5.1.9 Query Match as Variables ... 206
3.5.1.10 Work Time in Formula Columns ... 206
3.5.2 Wiki Markup in Formula Columns ... 207
3.5.2.1 Using Wiki Markup ... 207
3.5.2.2 Markup Options ... 209
3.5.2.3 Export ... 209
3.5.2.4 Examples ... 209
3.5.2.5 Wiki Markup Advanced Examples ... 211
3.5.3 Bundled Formulas ... 219
3.5.4 Expr Language ... 221
3.5.4.1 Language Components ... 221
3.5.4.2 Basic Constructs ... 222
3.5.4.3 Advanced Constructs ... 224
3.5.4.4 Expr Reference Guides ... 226
3.5.4.5 Expr Function Reference ... 226
3.5.4.6 Aggregate Function Reference ... 246
3.5.4.7 Expr Language Reference ... 255
3.5.4.8 Expr Pattern Matching ... 263
3.5.4.9 Expr Error Codes ... 265
3.6 Structured JQL ... 266
3.6.1 Get Started with S-JQL ... 266

3.6.2 Master S-JQL ... 266
3.6.3 S-JQL Cookbook ... 267
3.6.3.1 Find issues added to a structure ... 267
3.6.3.2 Quick Filter for JIRA Agile's (GreenHopper) Scrum Board to display only low-level issues in a structure ... 267
3.6.3.3 Retrieve all Epics in a certain status and all of their children ... 268
3.6.3.4 Find Test Cases associated with Stories in an active sprint ... 268
3.6.3.5 Find all issues that are blocking critical issues ... 269
3.6.3.6 Find all unassigned issues in a part of a project ... 269
3.6.3.7 Top-level view on unfinished parts of a project ... 269
3.6.3.8 Find violations of the rule "Tasks must be under Epics or Stories" ... 270
3.6.3.9 Find violations of the rule "An issue cannot be resolved if it has unresolved children" ... 270
3.6.3.10 Find issues that can be resolved because all their children are resolved ... 270
3.6.3.11 Get a view of a second (third, ...) level of the hierarchy ... 271
3.6.3.12 Get the contents of a folder ... 271
3.6.4 S-JQL Reference ... 272
3.6.4.1 Multiple instances of items ... 272
3.6.4.2 Constraints ... 272
3.6.4.3 Basic constraint ... 273
3.6.4.4 Negation ... 278
3.6.4.5 Relational constraint ... 278
3.6.4.6 Combining constraints with Boolean operators ... 284
3.6.4.7 Railroad diagrams ... 284
3.6.4.8 List of S-JQL keywords ... 286
3.6.5 structure() JQL function ... 293
3.6.5.1 Function arguments need to be quoted if they contain spaces or non-letters ... 294
3.6.5.2 Backward compatibility with structure() JQL function prior to Structure 2.4 ... 294
3.7 Columns and Views ... 295
3.7.1 Structure Columns ... 295
3.7.1.1 Issue Key Column ... 296
3.7.1.2 Summary Column ... 296
3.7.1.3 Field Columns ... 297
3.7.1.4 Icons Column ... 299
3.7.1.5 Progress Column ... 300
3.7.1.6 Images Column ... 314

3.7.1.7 Work Logged Column ... 315
3.7.1.8 Flags Column ... 317
3.7.1.9 Jira Actions Column ... 317
3.7.1.10 Sequential Index Column ... 318
3.7.1.11 Notes Column ... 319
3.7.1.12 Totals Columns ... 320
3.7.1.13 Time in Status Column ... 321
3.7.1.14 Status Category Column ... 325
3.7.1.15 Count Sub-Items Column ... 326
3.7.1.16 Count Leaves Column ... 327
3.7.1.17 Transition Dates ... 327
3.7.1.18 Query Match Column ... 328
3.7.1.19 Last Comment Column ... 330
3.7.2 Customizing Columns ... 331
3.7.2.1 Adding Columns ... 331
3.7.2.2 Configuring Columns ... 332
3.7.2.3 Removing Columns ... 332
3.7.2.4 Rearranging Columns ... 333
3.7.2.5 Resizing Columns and Autosize ... 333
3.7.3 Horizontal Scrolling ... 333
3.7.3.1 Pinned Columns ... 334
3.7.3.2 Saving Horizontal Scrolling Settings ... 335
3.7.4 Configuring View ... 336
3.7.4.1 Changing the View ... 337
3.7.4.2 Views Menu ... 337
3.7.4.3 Saving and Sharing Views ... 339
3.7.5 Managing Views ... 341
3.7.5.1 Locating a View ... 342
3.7.5.2 Changing View Settings ... 344
3.7.5.3 View Sharing and Permissions ... 346
3.7.5.4 Associating Views with Structures ... 348
3.7.5.5 Copying a View ... 348
3.7.5.6 Deleting a View ... 349
3.7.6 Displaying Full Cell Content ... 349
3.7.7 Two-Panel Mode ... 350

3.7.7.1 Viewing the Secondary Panel ... 350
3.7.7.2 Resizing the Secondary Panel ... 351
3.7.7.3 Structure Widget on Secondary Panel ... 351
3.7.7.4 Issue Clipboard ... 352
3.7.8 Full Screen Mode ... 353
3.7.8.1 Hide the Structure Toolbar ... 353
3.8 Managing Structures ... 354
3.8.1 Locating a Structure ... 356
3.8.1.1 Finding Structures by Name, Access Level or Owner ... 356
3.8.1.2 Finding a Structure by Its ID ... 357
3.8.2 Default Structure ... 357
3.8.3 Favorite Structures ... 357
3.8.3.1 Add/Remove Favorite Structures ... 358
3.8.3.2 Structure Popularity ... 358
3.8.4 Structure Details ... 359
3.8.4.1 Editing Structure Details ... 360
3.8.5 Customizing View Settings ... 360
3.8.5.1 Switching Between Default and Customized View Settings ... 361
3.8.5.2 Configuring Views Menu ... 361
3.8.5.3 Configuring Default View ... 362
3.8.6 Structure Permissions ... 362
3.8.6.1 Access Levels ... 362
3.8.6.2 Default Access ... 362
3.8.6.3 Permission Rules ... 363
3.8.7 Creating New Structures ... 365
3.8.8 Copying a Structure ... 365
3.8.8.1 Create a Copy ... 366
3.8.8.2 The New Structure ... 367
3.8.8.3 Copy vs. Clone ... 367
3.8.8.4 Copying Structure and Cloning Issues ... 368
3.8.9 Archiving a Structure ... 372
3.8.9.1 Archive a Structure ... 372
3.8.9.2 Unarchive a Structure ... 372
3.8.9.3 Searching for Archived structures ... 373
3.8.9.4 Synchronizers and Archiving ... 373

3.8.10 Deleting a Structure 374
3.8.11 Template Structures and Projects 374
3.8.11.1 Configuring Template Structures 375
3.8.11.2 Creating Issues and a Structure from Template 375
3.8.11.3 Template Projects 375
3.8.12 Viewing History of a Structure 375
3.8.12.1 Reading History View 376
3.8.12.2 Limitations of the History View 377
3.8.12.3 Printing a Previous Structure Version 377
3.8.12.4 Exporting a Previous Structure Version to Excel 377
3.8.13 Exporting Structures 377
3.8.13.1 Exporting Structures to PDF 377
3.8.13.2 Exporting Structure to XLS (Excel) 378
3.8.13.3 Printing Structure 380
3.8.14 Real-Time Collaboration 381
3.8.15 Structure Activity Stream 381
3.8.15.1 Available Filters 382
3.8.15.2 Reading Activity Stream 383
3.8.15.3 Activity Streams Performance 384
3.9 Keyboard Shortcuts 384
3.9.1 Keyboard Shortcuts (PC) 385
3.9.1.1 Navigation 385
3.9.1.2 Structure Views 385
3.9.1.3 Searching & Adding to Structure 386
3.9.1.4 Standard JIRA Actions 386
3.9.1.5 Changing Structure 387
3.9.1.6 Changing Issues 387
3.9.1.7 Selecting Issues 388
3.9.1.8 Advanced 389
3.9.2 Keyboard Shortcuts (Mac) 389
3.9.2.1 Navigation 389
3.9.2.2 Structure Views 390
3.9.2.3 Standard JIRA Actions 390
3.9.2.4 Changing Structure 391
3.9.2.5 Changing Issues 392

3.9.2.6 Selecting Issues ... 392
3.9.2.7 Advanced ... 393
3.9.3 Quick Action Lookup ... 393
3.10 Structure Gadget ... 394
3.10.1 Dashboard Gadget ... 394
3.10.1.1 Adding Structure Gadget to Dashboard ... 394
3.10.1.2 Configuring the Gadget ... 394
3.10.1.3 Customizing Gadget View ... 396
3.10.1.4 Using the Gadget ... 397
3.10.1.5 Open on Structure Board ... 397
3.10.2 Confluence Gadget ... 398
3.10.2.1 Adding a Structure Gadget to a Confluence Page ... 399
3.10.2.2 Using the Gadget ... 400
3.10.2.3 Customizing Gadget View ... 400
3.10.2.4 Open on Structure Board ... 402
3.11 Getting Help ... 403
3.11.1 Resources Within Structure ... 403
3.11.2 Contact Us ... 404
3.12 Integrations ... 404
3.12.1 Better PDF Exporter for Jira ... 405
3.12.1.1 Key benefits to using Better PDF Exporter with Structure ... 405
3.12.2 ScriptRunner ... 405
3.12.2.1 Use-cases ... 405
3.12.2.2 Automatically Remove Issues Based on JQL Query ... 406
3.12.2.3 Bulk Change Owners of Structures and Generators ... 408
3.12.2.4 Creating a New Structure Programmatically ... 410
3.12.2.5 Creating Generators ... 411
3.12.2.6 Show All Structure Boards and Corresponding Item Counts ... 415
3.12.2.7 Show Related Issues in a Separate Column ... 416
3.12.2.8 Show work logged per user and issue for a structure ... 418
3.12.2.9 Updating a field (ex. label) when checking all issues against a JQL query ... 420
3.12.3 Tempo Timesheets ... 420
3.12.3.1 Grouping by Tempo Accounts ... 421
3.12.4 Xporter ... 421
3.12.4.1 Key benefits to using Xporter with Structure ... 421

3.12.5 Xray Test Management for Jira ... 422
3.12.5.1 Key benefits to using Xray with Structure ... 422
3.12.6 Power Scripts for Jira ... 422
3.12.6.1 Key benefits to using Power Scripts with Structure ... 422
4 How-To Articles ... 423
4.1 Calculate the Cost to Complete Issues and Projects ... 423
4.1.1 Situation ... 423
4.1.2 Solution ... 423
4.1.2.1 Step 1: Add a Column for the Custom User Property ... 423
4.1.2.2 Step 2: Calculate the Cost to Finish Each Issue ... 424
4.1.3 Bonus - Make This Information Visible Outside of Structure ... 425
4.1.3.1 Add an Effector ... 425
4.1.3.2 Run the Effector ... 427
4.2 Update Assignees to Match Parent Issues ... 428
4.2.1 Situation ... 428
4.2.2 Solution ... 428
4.2.2.1 Step 1: Add a Formula Column to Capture the Parent's Assignee ... 429
4.2.2.2 Step 2: Add an Effector ... 430
4.2.2.3 Step 3: Run the Effector ... 431
4.3 Calculate Story Points for Epics and Write the Values to Jira ... 433
4.3.1 Situation ... 433
4.3.2 Solution ... 433
4.3.2.1 Step 1: Add a Formula Column to Calculated the Total Story Points ... 433
4.3.2.2 Step 2: Add an Effector ... 435
4.3.2.3 Step 3: Run the Effector ... 436
4.4 Backlog Grooming for Teams ... 438
4.4.1 Situation ... 438
4.4.2 Solution ... 438
4.4.2.1 Step 1 ... 438
4.4.2.2 Step 2 ... 438
4.4.2.3 Step 3 ... 439
4.4.2.4 Step 4 ... 439
4.4.2.5 Optional Enhancements: ... 440
4.5 Correcting Inconsistent or Incorrect Link Types with the Extend Generator ... 440

4.5.1 Situation ... 440
4.5.2 Solution ... 440
4.5.2.1 Step 1 ... 440
4.5.2.2 Step 2 ... 441
4.5.2.3 Step 3 ... 441
4.6 Resource Allocation ... 442
4.6.1 Situation ... 442
4.6.2 Solution ... 442
4.6.2.1 Step 1 ... 442
4.6.2.2 Step 2 ... 443
4.6.2.3 Step 3 ... 443
4.6.2.4 Step 4 ... 444
4.6.2.5 Optional Enhancements: ... 444
4.7 Enable SAFe (Scaled Agile Framework) with Structure ... 445
4.7.1 Requirements ... 445
4.7.2 Setup ... 445
4.7.3 Step-by-step guide ... 445
4.8 Creating an Advanced Formula Column ... 446
4.8.1 1. Write the Main Formula ... 446
4.8.2 2. Create a Temporary Column with Total Number of Bugs ... 447
4.8.3 3. Create a Temporary Column with Total Number of Issues ... 448
4.8.4 4. Define the Main Formula Variables ... 448
4.8.5 5. Choose an Appropriate Format ... 449
4.9 Track Overdue Tasks with Structure ... 450
4.9.1 Situation ... 450
4.9.2 Solution ... 450
4.9.2.1 Step 1: Build Your Structure ... 450
4.9.2.2 Step 2: Mark Overdue Tasks ... 451
4.9.2.3 Step 2: Aggregate Overdue Items for Epics, Initiatives, etc. ... 453
4.9.2.4 Step 4: Calculate the Ration of Overdue Tasks to All Tasks ... 455
5 Structure Administrator's Guide ... 458
5.1 Structure Best Practices ... 459
5.1.1 Initial Considerations ... 460
5.1.1.1 Organizations New to Jira ... 460

5.1.1.2 Organizations Established in Jira ... 460
5.1.2 Basic Terms & Principles ... 460
5.1.2.1 Automation ... 460
5.1.3 Things to Keep in Mind ... 461
5.1.4 Common Approaches to Building ... 462
5.1.4.1 Top Down ... 462
5.1.4.2 Focus on a Level ... 463
5.1.4.3 Visualize the Entire Scope ... 463
5.1.5 Performance Considerations ... 464
5.1.5.1 Automation ... 464
5.1.5.2 Progress, Totals and Formula Columns ... 465
5.1.5.3 Structure on Issue Page ... 465
5.1.5.4 S-JQL ... 465
5.1.5.5 Monitoring & Troubleshooting Structure Usage ... 465
5.1.6 Additional Resources ... 465
5.1.7 Monitoring and Troubleshooting Structure Usage ... 466
5.1.7.1 Reading the Performance Audit Log ... 466
5.2 Installing Structure ... 467
5.2.1 Migrating Data from Structure 2 to Structure 3 ... 468
5.2.1.1 Creating a Backup of Structure 2.x Data ... 468
5.2.1.2 Restoring Structure Data from 2.x Backup ... 468
5.2.1.3 After Data Migration ... 468
5.2.2 Memory Guidelines ... 469
5.2.2.1 Assessing Available Memory ... 469
5.2.2.2 Heap Memory Requirements ... 469
5.2.2.3 PermGen Memory Requirements ... 470
5.2.2.4 Changing Memory Parameters ... 470
5.2.2.5 Use 64-Bit Java ... 470
5.2.2.6 Physical Memory Requirements ... 470
5.2.3 Uninstalling and Reinstalling Structure ... 471
5.2.3.1 Uninstalling Structure ... 471
5.2.3.2 Reinstalling Structure ... 471
5.2.4 Upgrading and Downgrading ... 471
5.2.4.1 Upgrading ... 471
5.2.4.2 Downgrading ... 472

5.3 Setting Up Structure License 473
5.3.1 Setting Up Evaluation License 473
5.3.2 Licenses from ALM Works and from Atlassian 474
5.3.3 Purchasing a Commercial License 475
5.3.3.1 Purchasing from ALM Works 475
5.3.3.2 Purchasing from Atlassian 475
5.3.3.3 Purchasing from Resellers or Atlassian Experts 475
5.3.4 Migrating Licenses 476
5.3.5 Structure License Parameters 476
5.3.6 When Structure is Available for Free 477
5.3.7 License Maintenance and Expiration 477
5.3.7.1 Commercial License 477
5.3.7.2 Evaluation License 477
5.3.7.3 Read-only Mode 477
5.4 Selecting Structure-Enabled Projects 477
5.5 Global Permissions 478
5.5.1 Who Has Access to the Structure 478
5.5.2 Restricting User Access to Structure 478
5.5.3 Changing Permission to Create New Structures 479
5.5.4 Changing Permission to Access Automation 480
5.5.5 Changing Permissions to Configure and Run Effectors 480
5.6 Changing Structure Defaults 481
5.6.1 Initial Configuration 481
5.6.2 Changing Default Structure 482
5.6.2.1 Changing System-level Default Structure 482
5.6.2.2 Changing Project-level Default Structure 482
5.6.3 Changing Default View Settings 482
5.6.4 Changing Default Options for the Issue and Project Pages 483
5.7 Attribute Sensitivity Settings 483
5.7.1 How Does Attribute Sensitivity Work? 484
5.7.2 Setting the Attribute Sensitivity Mode 484
5.7.3 Assigning Specific Non-Sensitive Attributes 484
5.8 Structure Backup, Restore and Migration 485
5.8.1 Using Structure Backup 486

5.8.2 Backing Up Structure ... 486
5.8.3 Restoring Structure from Backup ... 487
5.8.3.1 Downgrading from Structure 5.0 or Later ... 487
5.8.4 Migrating Structures ... 488
5.8.4.1 Owners and Permissions ... 489
5.9 Automatic Structure Maintenance ... 490
5.9.1 Automatic Structure Maintenance ... 490
5.9.2 Maintenance Schedule ... 490
5.9.3 Maintenance Tasks ... 492
5.9.4 Running Maintenance Tasks Manually ... 493
5.10 Workflow Integration ... 494
5.10.1 Structure Workflow Validator ... 494
5.10.2 Structure Workflow Condition ... 496
5.11 Running Structure on Jira Data Center ... 496
5.11.1 Archived Projects and Structure ... 496
5.11.1.1 Archived Issues in Structure ... 496
5.11.1.2 Restoring an Archived Project ... 497
5.12 Anonymous Usage Statistics ... 497
5.12.1 Viewing Current Statistics ... 497
5.12.2 Turning Anonymous Usage Statistics On and Off ... 497
5.12.3 Structure Usage Statistics ... 498
5.13 Structure Files Location ... 498
5.13.1 Structure Files ... 498
5.13.1.1 $JIRA_HOME/structure ... 498
5.14 Turning Off Optional Features ... 498
5.15 Advanced Configuration and Dark Features ... 499
5.15.1 Setting Application Properties with the Structure Dark Features and Fine Tuning Interface ... 500
5.15.1.1 Guidelines for Adding/Removing Property and Values ... 500
5.15.2 Setting System Properties ... 501
5.15.2.1 Setting System Properties on Startup ... 501
5.15.2.2 Setting System Properties with Script Runner ... 501
5.15.3 Available Dark Features ... 502
5.15.3.1 Structure size limit ... 502
5.15.3.2 Structure Automation limits ... 502

5.15.3.3 Automation Defaults ... 502
5.15.3.4 Manual adjustments ... 503
5.15.3.5 Hidden Issue Links ... 503
5.15.3.6 Index Consistency Checks ... 503
5.15.3.7 Alternative initial values for project/type when creating an issue in dialog ... 503
5.15.3.8 Manage Permissions for Groups Without Belonging to Them ... 504
5.15.3.9 Resolved icon(green tick) ... 504
5.15.3.10 Structure Export Timeout ... 504
5.15.3.11 Time in Status - Refresh Period ... 505
5.15.3.12 Time Tracking on the Issue Page and Issue Details Page ... 505
5.15.3.13 Link-related changes made by Generators or Synchronizers ... 505
5.15.3.14 Synchronizers ... 506
5.15.4 Structured Predictions ... 507
5.15.4.1 Learn more about Structured Predictions ... 508
5.15.4.2 Configuring Structured Predictions ... 508
5.15.4.3 Using Structured Predictions ... 510
5.15.4.4 Structured Predictions FAQ ... 513
5.15.5 Synchronization ... 514
5.15.5.1 Import Synchronization ... 515
5.15.5.2 Export Synchronization ... 516
5.15.5.3 Installing a Synchronizer ... 517
5.15.5.4 Modifying Synchronizer ... 518
5.15.5.5 Removing Synchronizer ... 519
5.15.5.6 Turning Synchronizer On and Off ... 519
5.15.5.7 Running Resync ... 520
5.15.5.8 Synchronization and Permissions ... 521
5.15.5.9 Protection from Synchronizer Cycles ... 522
5.15.5.10 Bundled Synchronizers ... 523
5.15.5.11 Undo Synchronizer Actions ... 537
5.15.5.12 Copying Synchronizers ... 538
5.16 System Requirements ... 539
5.16.1 Atlassian Platform ... 539
5.16.2 Databases ... 540
5.16.3 Browsers ... 540
5.16.4 Server Requirements ... 540

5.16.5 Non-Conforming systems ... 541
5.17 Confluence Gadget - Admin Configuration ... 541
5.17.1 Adding Structure Gadget to Confluence Configuration ... 541
5.17.1.1 Troubleshooting ... 542
5.17.2 Setting Up CORS Filter in JIRA ... 542
5.17.2.1 Nginx Configuration Option ... 542
5.18 Clearing Structure Caches ... 543
5.19 Anonymizing Users ... 544
5.19.1 Review User's Structures ... 544
5.19.2 Transfer Ownership ... 545
5.20 Structure Troubleshooting ... 545
5.20.1 Collecting Support Zip ... 546
5.20.2 HAR Network Report ... 546
5.20.2.1 Collecting HAR Report with Google Chrome ... 546
5.20.3 Troubleshooting Synchronizers ... 547
5.20.3.1 Structure Audit Log ... 548
5.20.3.2 Log Files ... 548
5.20.4 Structured JQL Troubleshooting ... 549
5.20.5 Collecting Performance Snapshots ... 549
5.20.5.1 Download and install Atlas-Yourkit plugin. ... 549
5.20.5.2 Load Profiling Agent ... 550
5.20.5.3 Capturing CPU Performance Snapshot ... 551
5.20.5.4 Capturing Memory Snapshot ... 551
5.20.5.5 Sending the Snapshots to Support Team ... 551
5.20.5.6 After Profiling Session ... 551
5.20.5.7 Performance Snapshot Without Yourkit Plugin ... 551
5.20.6 Sending Files to Support Team ... 555
5.20.6.1 Attach to the Support Request in ALM Works Service Desk (Preferred) ... 555
5.20.6.2 Send Files by E-mail ... 555
5.20.6.3 Upload Files Directly to Our Server ... 556
5.20.7 Troubleshooting Performance and Stability Issues ... 556
5.20.7.1 Thread Dumps ... 556
5.20.7.2 Verbose Logging ... 557
5.20.7.3 Support Zip ... 557
5.20.7.4 Browser Console Log ... 557

5.20.7.5 HAR Report ... 558
5.20.7.6 Screenshots or Video ... 558
5.21 Migrate to Cloud ... 558
5.21.1 Creating a Migration Configuration ... 559
6 Frequently Asked Questions ... 561
6.1 Data Center Approved Apps FAQ ... 561
6.1.1 What are Data Center approved apps? ... 561
6.1.2 Didn’t Structure already work with Data Center? ... 562
6.1.3 Why has Structure made this change? ... 562
6.1.4 What are the criteria for being Data Center approved? ... 562
6.1.5 Why should I upgrade to the Data Center approved app version? ... 562
6.1.6 Why is the price different for Data Center approved apps? ... 563
6.1.7 Do I have to switch to the Data Center approved version? Can I continue to use the server app in Data Center? ... 563
6.2 Cannot Create an Issue With +Next Issue (+Sub-Issue) Because of the Required Fields ... 563
6.2.1 Question ... 563
6.2.2 Answer ... 563
6.2.2.1 1. Switch to the Create Issue Dialog ... 563
6.2.2.2 2. Add the Required Field to the View ... 563
6.3 Add-on Manager Says Structure Is Unlicensed ... 564
6.3.1 Question ... 564
6.3.2 Answer ... 564
6.4 No Check Mark Displayed for a Resolved Issue ... 564
6.4.1 Question ... 564
6.4.2 Answer ... 564
6.4.2.1 Problems Caused By Custom Workflows ... 565
6.4.2.2 Problems Caused By Manually Added "Unresolved" Resolution Value ... 565
6.5 Structure Won't Start ... 565
6.5.1 Question ... 565
6.5.2 Answer ... 565
6.5.2.1 Structure database cannot be created or opened; filesystem is read-only or full ... 566
6.5.2.2 Some of the required system plugins are disabled ... 566
6.5.2.3 Incomplete download or corrupt plugin JAR file ... 567
6.5.2.4 Incorrect Jira setup ... 567

6.6 After an Issue is Moved to Another Project, It Cannot Be Found in the Structure 567
6.6.1 Question 567
6.6.2 Answer 568
6.6.2.1 The New Project Is Not Structure Enabled 568
6.6.2.2 The Issue Was Removed Based on Automation Rules 568
6.7 User Cannot Access Structure 568
6.7.1 Question 568
6.7.2 Answer 568
6.7.2.1 If the user's permissions were just changed: 568
6.8 Where Do I Find the Jira Server ID? 568
6.9 Using Subtasks and Structure 569
6.9.1 Question 569
6.9.2 Answer 569
6.10 Difference from Sub-tasks 569
6.10.1 Question 569
6.10.2 Answer 569
6.11 Performance Considerations 569
6.12 How to restore the structure using History 570
6.13 Can a Structure be Exported to Microsoft Word? 570
6.14 Convert time data in Excel export to Jira format 570
6.14.1 How to Install 571
6.14.2 How to Use 571
7 Release Notes 572
7.1 Structure 6.4 Release Notes 572
7.1.1 Version Highlights 573
7.1.2 Changes in Detail 573
7.1.2.1 Fix Version and Sprint data 573
7.1.2.2 Photo and Name in User Fields 574
7.1.2.3 Additional Updates 574
7.1.3 Supported Versions 575
7.1.4 Installation and Upgrade 575
7.1.4.1 Installing Structure 575
7.1.4.2 Upgrading Structure 575
7.1.4.3 Upgrading from Structure 2.9–2.11 576

7.1.5 Enterprise Deployment Notes 576
7.1.5.1 Extender Performance Improvements 576
7.1.5.2 Confluence Gadget Updates 576
7.1.5.3 Other Improvements and Fixes 576
7.1.5.4 Testing on Staging Environment 577
7.1.6 API Changes in Structure 6.4 577
7.1.6.1 Minor Java API Release 577
7.1.6.2 Compatible Changes in the Java API 577
7.2 Structure 6.3 Release Notes 577
7.2.1 Version Highlights 578
7.2.2 Changes in Detail 578
7.2.2.1 Status Rollup Effector 578
7.2.2.2 Default Views 578
7.2.2.3 Group by Project Category and ScriptRunner Single Issue Picker 579
7.2.2.4 Improved Information Center 580
7.2.2.5 Additional Updates 580
7.2.3 Supported Versions 581
7.2.4 Installation and Upgrade 581
7.2.4.1 Installing Structure 581
7.2.4.2 Upgrading Structure 581
7.2.4.3 Upgrading from Structure 2.9–2.11 582
7.2.5 Enterprise Deployment Notes 582
7.2.5.1 Reading Custom Fields from the Index 582
7.2.5.2 Cluster Node Tracking in Structure Gadget in Confluence 582
7.2.5.3 More Reliable Export Under Load 583
7.2.5.4 Testing on Staging Environment 583
7.2.6 API Changes in Structure 6.3 583
7.2.6.1 Minor Java API Release 583
7.2.6.2 Compatible Changes in the Java API 584
7.2.7 API Changes in Structure 6.3.2 584
7.2.7.1 Minor Java API Release 584
7.2.7.2 Compatible Changes in the Java API 584
7.2.8 Structure 6.3.1 Release Notes 585
7.2.8.1 Patch Release 585
7.2.8.2 Supported Versions 585

7.2.8.3 Installation and Upgrade ... 585
7.2.9 Structure 6.3.2 Release Notes ... 586
7.2.9.1 Patch Release ... 587
7.2.9.2 Changes in Detail ... 587
7.2.9.3 Supported Versions ... 588
7.2.9.4 Installation and Upgrade ... 588
7.2.9.5 Enterprise Deployment Notes ... 589
7.3 Structure 6.2 Release Notes ... 590
7.3.1 Version Highlights ... 590
7.3.2 Changes in Detail ... 590
7.3.2.1 Effectors Can Write Values to Duration and Multi-Value Fields ... 590
7.3.2.2 Full-Screen Formula Editor ... 591
7.3.2.3 Last Comment Column ... 591
7.3.2.4 Additional Updates ... 592
7.3.3 Supported Versions ... 592
7.3.4 Installation and Upgrade ... 593
7.3.4.1 Installing Structure ... 593
7.3.4.2 Upgrading Structure ... 593
7.3.4.3 Upgrading from Structure 2.9–2.11 ... 593
7.3.5 Enterprise Deployment Notes ... 594
7.3.5.1 Faster Initial Loading of Structures from the Cache ... 594
7.3.5.2 Migration Improvements ... 594
7.3.5.3 Testing on Staging Environment ... 594
7.3.6 API Changes in Structure 6.2 ... 595
7.3.6.1 Minor Java API Release ... 595
7.3.6.2 Compatible Changes in the Java API ... 595
7.3.6.3 New REST API for App Configuration ... 596
7.3.7 Structure 6.2.1 Release Notes ... 596
7.3.7.1 Patch Release ... 596
7.3.7.2 Supported Versions ... 596
7.3.7.3 Installation and Upgrade ... 597
7.4 Structure 6.1 Release Notes ... 598
7.4.1 Version Highlights ... 598
7.4.2 Changes in Detail ... 598
7.4.2.1 Rich Text Editor for Memos ... 598

7.4.2.2 Query Match Columns ... 599
7.4.2.3 JQL, S-JQL and Text Filters in Formulas ... 599
7.4.2.4 Jira Anonymization ... 600
7.4.2.5 Additional Updates ... 600
7.4.3 Installation and Upgrade ... 600
7.4.3.1 Installing Structure ... 600
7.4.3.2 Upgrading Structure ... 600
7.4.3.3 Upgrading from Structure 2.9–2.11 ... 601
7.4.4 Enterprise Deployment Notes ... 601
7.4.5 API Changes in Structure 6.1 ... 601
7.4.5.1 Minor Java API Release ... 601
7.4.5.2 Compatible Changes in the Java API ... 602
7.4.6 Structure 6.1.1 Release Notes ... 602
7.4.6.1 Patch Release ... 602
7.4.6.2 Installation and Upgrade ... 602
7.4.7 Structure 6.1.2 Release Notes ... 603
7.4.7.1 Patch Release ... 604
7.4.7.2 Installation and Upgrade ... 604
7.5 Structure 6.0 Release Notes ... 605
7.5.1 Version Highlights ... 605
7.5.2 Changes in Detail ... 606
7.5.2.1 Effectors ... 606
7.5.2.2 Permissions Per User ... 606
7.5.2.3 API Changes ... 606
7.5.2.4 Attribute Sensitivity Settings ... 607
7.5.2.5 Improved Rank Sorting ... 607
7.5.3 Supported Versions ... 607
7.5.4 Installation and Upgrade ... 608
7.5.4.1 Installing Structure ... 608
7.5.4.2 Upgrading Structure ... 608
7.5.4.3 Upgrading from Structure 2.9–2.11 ... 608
7.5.5 Enterprise Deployment Notes ... 609
7.5.5.1 Changes to the Attribute System ... 609
7.5.5.2 Effectors ... 609
7.5.6 Structure 6 API Changes ... 610

7.5.6.1 State of the API ... 610
7.5.6.2 Conceptual API Changes ... 610
7.5.6.3 Detailed API Changes ... 611
7.5.6.4 Conceptual SPI Changes ... 612
7.5.6.5 Detailed SPI Changes ... 613
7.5.6.6 Effector APIs and SPIs ... 614
7.5.7 Structure for Jira New Server Pricing ... 615
7.5.8 Structure 6.0.1 Release Notes ... 616
7.5.8.1 Patch Release ... 616
7.5.8.2 Installation ... 616
7.5.8.3 Upgrade ... 616
7.5.8.4 Enterprise Deployment Notes ... 617
7.6 Structure 5.6 Release Notes ... 617
7.6.1 Version Highlights ... 617
7.6.2 Changes in Detail ... 617
7.6.2.1 Column Pinning ... 617
7.6.2.2 Clearing Structure Caches ... 618
7.6.2.3 Additional Updates ... 618
7.6.3 Supported Versions ... 618
7.6.4 Installation and Upgrade ... 619
7.6.4.1 Installing Structure ... 619
7.6.4.2 Upgrading Structure ... 619
7.6.5 Enterprise Deployment Notes ... 619
7.6.5.1 Lucene Searcher Flushing ... 620
7.6.5.2 Reduced HTTP Thread Pool Usage ... 620
7.6.5.3 Clearing Caches and Hard Refresh ... 621
7.6.5.4 Testing on Staging Environment ... 621
7.6.6 API Changes in Structure 5.6 ... 621
7.6.6.1 Minor Java API Release ... 621
7.6.6.2 Compatible Changes in the Java API ... 621
7.6.7 Structure 5.6.1 Release Notes ... 622
7.6.7.1 Patch Release ... 622
7.6.7.2 Installation ... 622
7.6.7.3 Upgrade ... 622
7.6.8 Structure 5.6.2 Release Notes ... 623

7.6.8.1 Patch Release ... 623
7.6.8.2 Installation ... 623
7.6.8.3 Upgrade ... 623
7.6.8.4 4. Enterprise Deployment Notes ... 624
7.6.9 Structure 5.6.3 Release Notes ... 624
7.6.9.1 Patch Release ... 624
7.6.9.2 Installation ... 624
7.6.9.3 Upgrade ... 624
7.6.9.4 4. Enterprise Deployment Notes ... 625
7.6.10 Structure 5.6.4 Release Notes ... 625
7.6.10.1 Patch Release ... 625
7.6.10.2 Installation ... 625
7.6.10.3 Upgrade ... 625
7.7 Structure 5.5 Release Notes ... 626
7.7.1 Version Highlights ... 626
7.7.2 Changes in Detail ... 626
7.7.2.1 Transformation Timeout ... 626
7.7.2.2 Backup/Restore/Migration Improvements ... 626
7.7.3 Supported Versions ... 627
7.7.4 Installation and Upgrade ... 627
7.7.4.1 Installing Structure ... 627
7.7.4.2 Upgrading Structure ... 627
7.7.5 Enterprise Deployment Notes ... 628
7.7.6 API Changes in Structure 5.5 ... 628
7.7.6.1 Minor Java API Release ... 628
7.7.6.2 Compatible Changes in the Java API ... 628
7.7.7 Structure 5.5.1 Release Notes ... 628
7.7.7.1 Patch Release ... 629
7.7.7.2 Installation ... 629
7.7.7.3 Upgrade ... 629
7.8 Structure 5.4 Release Notes ... 629
7.8.1 Version Highlights ... 630
7.8.2 Changes in Detail ... 630
7.8.2.1 Time in Status ... 630
7.8.2.2 Memo Items ... 630

7.8.2.3 Status Category Column ... 631
7.8.2.4 Additional Updates ... 631
7.8.3 Supported Versions ... 632
7.8.4 Installation and Upgrade ... 632
7.8.4.1 Installing Structure ... 632
7.8.4.2 Upgrading Structure ... 632
7.8.5 Enterprise Deployment Notes ... 633
7.8.5.1 Periodical Cleaning of the JQL Query Literals Cache ... 633
7.8.5.2 Time in Status Column ... 633
7.8.5.3 Memo Item Type ... 633
7.8.5.4 Testing on a Staging Environment ... 633
7.8.6 API Changes in Structure 5.4 ... 634
7.8.6.1 Minor Java API Release ... 634
7.8.6.2 Compatible Changes in the Java API ... 634
7.8.7 Structure 5.4.1 Release Notes ... 634
7.8.7.1 Patch Release ... 635
7.8.7.2 Installation ... 635
7.8.7.3 Upgrade ... 635
7.9 Structure 5.3 Release Notes ... 635
7.9.1 Version Highlights ... 635
7.9.2 Changes in Detail ... 636
7.9.2.1 Version Name Grouper ... 636
7.9.2.2 Hide Quick Transformations panel ... 637
7.9.2.3 Mark Manual Adjustments ... 637
7.9.2.4 Additional Updates ... 638
7.9.3 Supported Versions ... 638
7.9.4 Installation and Upgrade ... 638
7.9.4.1 Installing Structure ... 638
7.9.4.2 Upgrading Structure ... 639
7.9.5 Enterprise Deployment Notes ... 639
7.9.5.1 Core Attribute Changes ... 639
7.9.5.2 Custom Field Scope Checks ... 639
7.9.5.3 Testing on Staging Environment ... 639
7.9.6 API Changes in Structure 5.3 ... 640
7.9.6.1 Minor Java API Release ... 640

7.9.6.2 Compatible Changes in the Java API ... 640
7.10 Structure 5.2 Release Notes ... 641
7.10.1 Version Highlights ... 641
7.10.2 Changes in Detail ... 642
7.10.2.1 Support for Jira 8.0 ... 642
7.10.2.2 Smoother Transition from Structure Gadget to Structure Board ... 642
7.10.2.3 Manage Structure in Structure Selection Menu ... 642
7.10.3 Supported Versions ... 643
7.10.4 Installation and Upgrade ... 643
7.10.4.1 Installing Structure ... 643
7.10.4.2 Upgrading Structure ... 643
7.10.5 Enterprise Deployment Notes ... 644
7.10.6 API Changes in Structure 5.2 ... 644
7.10.6.1 Minor Java API Release ... 644
7.10.6.2 Compatible Changes in the Java API ... 644
7.10.6.3 JavaScript API for adding item details support ... 645
7.10.7 Structure 5.2.1 Release Notes ... 645
7.10.7.1 Patch Release ... 645
7.10.7.2 Installation ... 646
7.10.7.3 Upgrade ... 646
7.10.7.4 Enterprise Deployment Notes ... 646
7.10.8 Structure 5.2.2 Release Notes ... 647
7.10.8.1 Patch Release ... 647
7.10.8.2 Installation ... 647
7.10.8.3 Upgrade ... 647
7.10.8.4 4. Enterprise Deployment Notes ... 648
7.11 Structure 5.1 Release Notes ... 648
7.11.1 Version Highlights ... 648
7.11.2 Changes in Detail ... 648
7.11.2.1 Wiki Markup ... 648
7.11.2.2 Inactive User Flag ... 649
7.11.2.3 Admin Interface for Dark Features ... 649
7.11.2.4 Notable Fixes and Improvements ... 650
7.11.3 Supported Versions ... 650
7.11.4 Installation and Upgrade ... 651

7.11.4.1 Installing Structure 651
7.11.4.2 Upgrading Structure 651
7.11.5 Enterprise Deployment Notes 651
7.11.5.1 Structure Index Monitor Field data center 651
7.11.5.2 Offloading Change Stream Writing to a Background Thread data centerexperimental 652
7.11.5.3 Lucene and Jira Search Integration 652
7.11.5.4 Bulk Loading Rows from the Database 653
7.11.5.5 Testing on Staging Environment 653
7.11.6 API Changes in Structure 5.1 653
7.11.6.1 Minor Java API Release 653
7.11.6.2 Compatible Changes in the Java API 653
7.11.7 Structure 5.1.1 Release Notes 654
7.11.7.1 Patch Release 654
7.11.7.2 Installation 654
7.11.7.3 Upgrade 654
7.12 Structure 5.0 Release Notes 655
7.12.1 Version Highlights 655
7.12.2 Changes in Detail 655
7.12.2.1 Manual Adjustments 655
7.12.2.2 Horizontal Scrolling 656
7.12.2.3 Work Logged by User, Group or Role 656
7.12.2.4 Automatic Maintenance Scheduling with Crontab 656
7.12.2.5 Other Changes 656
7.12.3 Supported Versions 656
7.12.4 Installation and Upgrade 656
7.12.4.1 Installing Structure 656
7.12.4.2 Upgrading Structure 657
7.12.5 Enterprise Deployment Notes 657
7.12.5.1 Controlling Manual Adjustments 657
7.12.5.2 New Database Table for Manual Adjustments 658
7.12.5.3 Backup Format Version Change 658
7.12.5.4 Sub-Task Extender Performance Improvement 658
7.12.5.5 Testing on Staging Environment 658
7.12.6 API Changes in Structure 5.0 659
7.12.6.1 Minor Java API Release 659

7.12.6.2 Compatible Changes in the Java API ... 659
7.12.6.3 Incompatible changes in @Internal interfaces ... 660
7.13 Structure 4.6 Release Notes ... 660
7.13.1 Version Highlights ... 660
7.13.2 Changes in Detail ... 661
7.13.2.1 Automation Timeout ... 661
7.13.2.2 Inline Status Editing ... 661
7.13.2.3 Notable Fixes and Improvements ... 661
7.13.3 Supported Versions ... 662
7.13.4 Installation and Upgrade ... 662
7.13.4.1 Installing Structure ... 662
7.13.4.2 Upgrading Structure ... 662
7.13.5 Enterprise Deployment Notes ... 663
7.13.5.1 Automation Timeout ... 663
7.13.5.2 Synchronizers on Jira Data Center ... 663
7.13.5.3 Lucene Index Monitoring ... 663
7.13.5.4 Old Row Manager Implementation Removed ... 664
7.13.5.5 Testing on Staging Environment ... 664
7.13.6 API Changes in Structure 4.6 ... 664
7.13.6.1 Minor Java API Release ... 664
7.13.6.2 Compatible Changes in the Java API ... 664
7.13.7 Structure 4.6.1 Release Notes ... 665
7.13.7.1 Patch Release ... 665
7.13.7.2 Installation ... 666
7.13.7.3 Upgrade ... 666
7.13.7.4 Enterprise Deployment Notes ... 666
7.13.8 Structure 4.6.3 Release Notes ... 666
7.13.8.1 Patch Release ... 666
7.13.8.2 Installation ... 667
7.13.8.3 Upgrade ... 667
7.13.8.4 Enterprise Deployment Notes ... 667
7.13.9 Structure 4.6.5 Release Notes ... 668
7.13.9.1 1. Patch Release ... 668
7.14 Structure 4.5 Release Notes ... 669
7.14.1 Version Highlights ... 670

7.14.2 Changes in Detail ... 670
7.14.2.1 Duplicate Items Highlighting ... 670
7.14.2.2 Notable Fixes and Improvements ... 670
7.14.3 Supported Versions ... 671
7.14.4 Installation and Upgrade ... 671
7.14.4.1 Installing Structure ... 671
7.14.4.2 Upgrading Structure ... 671
7.14.5 Enterprise Deployment Notes ... 671
7.14.5.1 Extender Performance Optimizations ... 672
7.14.5.2 Other Changes ... 672
7.14.5.3 Testing on Staging Environment ... 672
7.14.6 API Changes in Structure 4.5 ... 672
7.14.6.1 Minor Java API Release ... 672
7.14.6.2 Compatible Changes in the Java API ... 673
7.14.7 Structure 4.5.1 Release Notes ... 673
7.14.7.1 Patch Release ... 674
7.14.7.2 Installation ... 674
7.14.7.3 Upgrade ... 674
7.14.7.4 Enterprise Deployment Notes ... 674
7.15 Structure 4.4 Release Notes ... 675
7.15.1 Version Highlights ... 675
7.15.2 Changes in Detail ... 675
7.15.2.1 Changing issue status in a structure ... 675
7.15.2.2 Totals in double grid ... 675
7.15.2.3 Changes to Performance Audit Log ... 676
7.15.2.4 Notable fixes and improvements ... 676
7.15.3 Supported Versions ... 676
7.15.4 Installation and Upgrade ... 676
7.15.4.1 Installing Structure ... 676
7.15.4.2 Upgrading Structure ... 676
7.15.5 Enterprise Deployment Notes ... 677
7.15.5.1 Index Monitoring Improvements ... 677
7.15.5.2 New Table for Generic Items ... 677
7.15.5.3 Testing on Staging Environment ... 677
7.15.6 API Changes in Structure 4.4 ... 678

7.15.6.1 Minor Java API Release ... 678
7.15.6.2 Compatible Changes in the Java API ... 678
7.16 Structure 4.3 Release Notes ... 678
7.16.1 Version Highlights ... 679
7.16.2 Changes in Detail ... 679
7.16.2.1 New Aggregate functions and modifiers ... 679
7.16.2.2 Using notes column in formulas ... 680
7.16.2.3 Filtering by sprint status ... 680
7.16.2.4 Security patch ... 680
7.16.2.5 Notable fixes and improvements ... 680
7.16.3 Supported Versions ... 680
7.16.4 Installation and Upgrade ... 681
7.16.4.1 Installing Structure ... 681
7.16.4.2 Upgrading Structure ... 681
7.16.5 Enterprise Deployment Notes ... 681
7.16.5.1 Formulas and Attribute Subsystem ... 681
7.16.5.2 Time Tracking Section Changes ... 682
7.16.5.3 Automation Engine Improvements ... 682
7.16.5.4 Testing on Staging Environment ... 682
7.16.6 API Changes in Structure 4.3 ... 682
7.16.6.1 Minor Java API Release ... 682
7.16.6.2 Compatible Changes in the Java API ... 683
7.17 Structure 4.2 Release Notes ... 684
7.17.1 Version Highlights ... 684
7.17.2 Changes in Detail ... 685
7.17.2.1 Formula Language Improvements ... 685
7.17.2.2 Custom Period in the Work Logged Column ... 685
7.17.2.3 Full Content Hover Box ... 685
7.17.2.4 Notable fixes and improvements ... 686
7.17.3 Supported Versions ... 686
7.17.4 Installation and Upgrade ... 686
7.17.4.1 Installing Structure ... 686
7.17.4.2 Upgrading Structure ... 687
7.17.5 Enterprise Deployment Notes ... 687
7.17.5.1 Events, Re-indexing and Experimental Feature ... 687

7.17.5.2 Cancelling Structure Restore ... 688
7.17.5.3 Formulas and Attribute Subsystem ... 688
7.17.5.4 Testing on Staging Environment ... 688
7.17.6 API Changes in Structure 4.2 ... 689
7.17.6.1 Minor Java API Release ... 689
7.17.6.2 Compatible Changes in the Java API ... 689
7.17.6.3 Item Type API Example ... 690
7.18 Structure 4.1 Release Notes ... 690
7.18.1 Version Highlights ... 691
7.18.2 Changes in Detail ... 691
7.18.2.1 Notes Column ... 691
7.18.2.2 Grouping by Version and Sprint names ... 691
7.18.2.3 Full-Screen Mode ... 692
7.18.2.4 Structure Size Limit ... 692
7.18.2.5 Security Patch ... 692
7.18.2.6 Notable fixes and improvements ... 693
7.18.3 Supported Versions ... 693
7.18.4 Installation and Upgrade ... 693
7.18.4.1 Installing Structure ... 693
7.18.4.2 Upgrading Structure ... 693
7.18.5 Enterprise Deployment Notes ... 694
7.18.5.1 About Structure Size Limit ... 694
7.18.5.2 New Table for Per-Structure, Per-Item Properties ... 694
7.18.5.3 Testing on Staging Environment ... 694
7.18.6 API Changes in Structure 4.1 ... 695
7.18.6.1 Minor Java API Release ... 695
7.18.6.2 Compatible Changes in the Java API ... 695
7.18.7 Structure 4.1.1 Release Notes ... 696
7.18.7.1 Patch Release ... 696
7.18.7.2 Installation ... 696
7.18.7.3 Upgrade ... 696
7.18.8 Structure 4.1.2 Release Notes ... 697
7.18.8.1 Patch Release ... 697
7.18.8.2 Installation ... 697
7.18.8.3 Upgrade ... 697

7.19 Structure 4.0 Release Notes 698
7.19.1 Version Highlights 698
7.19.2 Changes in Detail 699
7.19.2.1 Formula Column 699
7.19.2.2 Work Logged by User 700
7.19.2.3 Making Static Copy of a Structure 700
7.19.2.4 Exporting Structures to Other Add-ons 700
7.19.2.5 Notable Fixes and Improvements 701
7.19.3 Supported Versions 701
7.19.4 Installation and Upgrade 701
7.19.4.1 Installing Structure 701
7.19.4.2 Upgrading Structure 701
7.19.5 Enterprise Deployment Notes 702
7.19.5.1 Formula Column Notes 702
7.19.5.2 Access to User Properties 702
7.19.5.3 Copying and Cloning Structures 702
7.19.5.4 Backup 702
7.19.5.5 Testing on Staging Environment 702
7.19.6 API Changes in Structure 4.0 703
7.19.6.1 Minor Java API Release 703
7.19.6.2 Compatible Changes in the Java API 703
7.19.6.3 Web Items 703
7.20 Structure 3.6 Release Notes 704
7.20.1 Version Highlights 704
7.20.2 Changes in Detail 704
7.20.2.1 Grouping by Account 704
7.20.2.2 Support for Portfolio 2.2.3 and 2.2.4 705
7.20.2.3 Notable Fixes and Improvements 705
7.20.3 Supported Versions 705
7.20.4 Installation and Upgrade 705
7.20.4.1 Installing Structure 705
7.20.4.2 Upgrading Structure 706
7.20.5 Enterprise Deployment Notes 706
7.20.5.1 Backup 706
7.20.5.2 Testing on Staging Environment 706

7.20.6 API Changes in Structure 3.6 707
7.20.6.1 Minor Java API Release 707
7.20.6.2 Compatible Changes in the Java API 707
7.21 Structure 3.5 Release Notes 708
7.21.1 Version Highlights 708
7.21.2 Changes in Detail 709
7.21.2.1 Grouping by Issue Links 709
7.21.2.2 Improved Filtering in Structure Gadget 709
7.21.2.3 Bidirectional Issue Link Extender 710
7.21.2.4 Default Level Limit for Extenders 710
7.21.2.5 Security Patch 710
7.21.2.6 Notable Fixes and Improvements 710
7.21.3 Supported Versions 711
7.21.4 Installation and Upgrade 711
7.21.4.1 Installing Structure 711
7.21.4.2 Upgrading Structure 711
7.21.5 Enterprise Deployment Notes 712
7.21.5.1 Security Patches 712
7.21.5.2 Plugin Start-Up Sequence 712
7.21.5.3 Automation Engine Locking 712
7.21.5.4 Asynchronous Index Writing 713
7.21.5.5 Testing on Staging Environment 713
7.21.6 API Changes in Structure 3.5 714
7.21.6.1 Minor Java API Release 714
7.21.6.2 Compatible Changes in the Java API 714
7.21.7 Structure 3.5.1 Release Notes 714
7.21.7.1 Patch Release 714
7.21.7.2 Installation 715
7.21.7.3 Upgrade 715
7.21.8 Structure 3.5.2 Release Notes 715
7.21.8.1 Patch Release 716
7.21.8.2 Installation 716
7.21.8.3 Upgrade 716
7.22 Structure 3.4 Release Notes 716
7.22.1 Version Highlights 717

7.22.2 Changes in Detail ... 717
7.22.2.1 Quick Transformations ... 717
7.22.2.2 Integration with Portfolio for JIRA ... 717
7.22.2.3 Issue Details Layout on Project Page ... 718
7.22.2.4 Grouping by Text Field ... 718
7.22.2.5 Performance Improvements ... 718
7.22.2.6 Public API ... 718
7.22.2.7 Notable Fixes and Improvements ... 719
7.22.3 Supported Versions ... 719
7.22.3.1 Structure.Pages Upgrade Required ... 719
7.22.3.2 Structure.Testy Upgrade Required ... 719
7.22.4 Installation and Upgrade ... 719
7.22.4.1 Installing Structure ... 719
7.22.4.2 Upgrading Structure ... 720
7.22.5 Enterprise Deployment Notes ... 720
7.22.5.1 New implementation of RowManager component ... 720
7.22.5.2 Indexing on Data Center ... 721
7.22.5.3 Improved Start-up Sequence ... 721
7.22.5.4 Performance Improvements ... 721
7.22.5.5 Testing on Staging Environment ... 721
7.22.6 Structure 3.4.1 Release Notes ... 722
7.22.6.1 Patch Release ... 722
7.22.6.2 Installation ... 722
7.22.6.3 Upgrade ... 723
7.22.6.4 Enterprise Deployment Notes ... 723
7.23 Structure 3.3 Release Notes ... 723
7.23.1 Version Highlights ... 724
7.23.2 Changes in Detail ... 724
7.23.2.1 Structure Data Migration ... 724
7.23.2.2 Search for folders and other non-issue items with S-JQL ... 724
7.23.2.3 Special filter for removing duplicates from using Inserter and Extender ... 724
7.23.2.4 Maintenance task to clear old change history ... 725
7.23.2.5 Swapping panels in two-panel mode ... 725
7.23.2.6 JIRA 7.2 compatibility ... 725
7.23.2.7 Notable Fixes and Improvements ... 725

7.23.3 Supported Versions ... 725
7.23.3.1 Structure.Pages Upgrade Required ... 726
7.23.3.2 Structure.Testy Upgrade Required ... 726
7.23.4 Installation and Upgrade ... 726
7.23.4.1 Installing Structure ... 726
7.23.4.2 Upgrading Structure ... 726
7.23.5 Structure 3.3.1 Release Notes ... 726
7.23.5.1 Patch Release ... 727
7.23.5.2 Installation ... 727
7.23.5.3 Upgrade ... 727
7.23.6 Structure 3.3.2 Release Notes ... 728
7.23.6.1 Patch Release ... 728
7.23.6.2 Installation ... 728
7.23.6.3 Upgrade ... 728
7.23.7 Structure 3.3.3 Release Notes ... 729
7.23.7.1 Patch Release ... 729
7.23.7.2 Installation ... 729
7.23.7.3 Upgrade ... 729
7.23.7.4 Enterprise Deployment Notes ... 730
7.23.8 Structure 3.3.4 Release Notes ... 730
7.23.8.1 Patch Release ... 730
7.23.8.2 Installation ... 731
7.23.8.3 Upgrade ... 731
7.23.9 Structure 3.3.5 Release Notes ... 731
7.23.9.1 Patch Release ... 731
7.23.9.2 Installation ... 732
7.23.9.3 Upgrade ... 732
7.24 Structure 3.2 Release Notes ... 732
7.24.1 Version Highlights ... 733
7.24.2 Changes in Detail ... 733
7.24.2.1 Backup and Restore ... 733
7.24.2.2 Structure on the Project Page ... 733
7.24.2.3 Sequential Index Column ... 734
7.24.2.4 Quick Action Lookup ... 734
7.24.2.5 Structure Gadget in Confluence ... 734

7.24.2.6 Notable Fixes and Other Improvements ... 734
7.24.3 Supported Versions ... 735
7.24.3.1 Structure.Pages Upgrade Required ... 735
7.24.3.2 Structure.Testy Upgrade Required ... 735
7.24.4 Installation and Upgrade ... 735
7.24.4.1 Installing Structure ... 735
7.24.4.2 Upgrading Structure ... 735
7.24.5 Structure 3.2.1 Release Notes ... 736
7.24.5.1 Patch Release ... 736
7.24.5.2 Installation ... 736
7.24.5.3 Upgrade ... 737
7.24.6 Structure 3.2.2 Release Notes ... 737
7.24.6.1 Patch Release ... 737
7.24.6.2 Installation ... 738
7.24.6.3 Upgrade ... 738
7.24.7 Structure 3.2.3 Release Notes ... 738
7.24.7.1 Patch Release ... 738
7.24.7.2 Installation ... 739
7.24.7.3 Upgrade ... 739
7.25 Structure 3.1 Release Notes ... 739
7.25.1 Incremental Update ... 740
7.25.1.1 Structure.Pages Release ... 740
7.25.1.2 Notable Fixes and Improvements ... 740
7.25.2 Supported Versions ... 740
7.25.2.1 Structure.Testy Upgrade Required ... 740
7.25.3 Installation and Upgrade ... 741
7.25.3.1 Installing Structure ... 741
7.25.3.2 Upgrading Structure ... 741
7.25.4 Structure 3.1.1 Release Notes ... 741
7.25.4.1 Patch Release ... 741
7.25.4.2 Installation ... 742
7.25.4.3 Upgrade ... 742
7.26 Structure 3.0 Release Notes ... 742
7.26.1 Structure 3 – a Different Experience ... 743
7.26.2 Structure 3 Highlights ... 743

7.26.3 Notes on Structure 2 Features ... 743
7.26.3.1 Synchronizers ... 743
7.26.3.2 Other Changes ... 744
7.26.4 Supported Versions ... 744
7.26.4.1 Browser Support ... 744
7.26.5 Compatibility Issues ... 744
7.26.5.1 Other JIRA Plugins ... 744
7.26.5.2 REST API ... 745
7.26.5.3 Remote Gadget Not Available ... 745
7.26.6 Changes in API ... 745
7.26.7 Installation and Upgrade ... 745
7.26.7.1 Installing Structure for the first time ... 745
7.26.7.2 Upgrading from Structure 2.9–2.11 ... 746
7.26.7.3 Upgrading from preview versions of Structure 3.0 ... 746
7.26.7.4 Upgrading "Global Structure" ... 747
7.26.7.5 Downgrading ... 747
7.26.8 Feedback Is Welcome! ... 747
7.26.9 Structure 3.0.1 Release Notes ... 747
7.26.9.1 Patch Release ... 747
7.26.9.2 Installation ... 748
7.26.9.3 Upgrade ... 749
7.26.10 Structure 3.0.2 Release Notes ... 749
7.26.10.1 Patch Release ... 749
7.26.10.2 Installation ... 750
7.26.10.3 Upgrade ... 750
7.26.11 Preview Version Release Notes ... 750
7.26.11.1 Structure 3.0 RC1 Release Notes ... 750
7.26.11.2 Structure 3.0 Beta 2 Release Notes ... 755
7.26.11.3 Structure 3.0 Beta 1 Release Notes ... 759
7.27 (HIDDEN) Release Notes (Structure 1-2) ... 761
7.27.1 Structure 2.9 Release Notes ... 763
7.27.1.1 Version Highlights ... 763
7.27.1.2 Changes in Detail ... 763
7.27.1.3 Supported JIRA Versions ... 764
7.27.1.4 Changes for Developers ... 764

7.27.1.5 Installation 765
7.27.1.6 Upgrading from a Previous Version 765
7.27.1.7 API Changes in Structure 2.9 765
7.27.2 Structure 2.8 Release Notes 766
7.27.2.1 Version Highlights 767
7.27.2.2 Changes in Detail 767
7.27.2.3 Supported JIRA Versions 768
7.27.2.4 Changes for Developers 768
7.27.2.5 Installation 768
7.27.2.6 Upgrading from a Previous Version 768
7.27.2.7 Structure 2.8.1 Release Notes 768
7.27.2.8 Structure 2.8.2 Release Notes 769
7.27.2.9 Structure 2.8.3 Release Notes 770
7.27.2.10 API Changes in Structure 2.8 770
7.27.3 Structure 2.7 Release Notes 771
7.27.3.1 Version Highlights 771
7.27.3.2 Changes in Detail 772
7.27.3.3 Supported JIRA Versions 772
7.27.3.4 Changes for Developers 772
7.27.3.5 Installation 773
7.27.3.6 Upgrading from a Previous Version 773
7.27.3.7 API Changes in Structure 2.7 773
7.27.4 Structure 2.6 Release Notes 774
7.27.4.1 Version Highlights 774
7.27.4.2 Changes in Detail 774
7.27.4.3 Supported JIRA Versions 775
7.27.4.4 Changes for Developers 775
7.27.4.5 Installation 775
7.27.4.6 Upgrading from a Previous Version 775
7.27.4.7 Structure 2.6.1 Release Notes 776
7.27.4.8 API Changes in Structure 2.6 777
7.27.5 Structure 2.5 Release Notes 777
7.27.5.1 Version Highlights 777
7.27.5.2 Changes in Detail 777
7.27.5.3 Supported JIRA Versions 780

7.27.5.4 Changes for Developers ... 780
7.27.5.5 Installation ... 780
7.27.5.6 Upgrading from a Previous Version ... 780
7.27.5.7 Detailed Change List ... 781
7.27.5.8 Structure 2.5.1 Release Notes ... 781
7.27.5.9 Structure 2.5.2 Release Notes ... 782
7.27.5.10 Structure 2.5.3 Release Notes ... 783
7.27.5.11 API Changes in Structure 2.5 ... 784
7.27.6 Structure 2.4 Release Notes ... 785
7.27.6.1 Structured JQL ... 785
7.27.6.2 Other Changes ... 786
7.27.6.3 Supported JIRA Versions ... 786
7.27.6.4 Installation ... 787
7.27.6.5 Upgrading from a Previous Version ... 787
7.27.6.6 Structure 2.4.1 Release Notes ... 787
7.27.6.7 Structure 2.4.2 Release Notes ... 789
7.27.6.8 Structure 2.4.3 Release Notes ... 790
7.27.6.9 API Changes in Structure 2.4 ... 791
7.27.7 Structure 2.3 Release Notes ... 792
7.27.7.1 JIRA 6 Support ... 792
7.27.7.2 Other Changes ... 792
7.27.7.3 Supported JIRA Versions ... 792
7.27.7.4 Installation ... 793
7.27.7.5 Upgrading from a Previous Version ... 793
7.27.7.6 API Changes in Structure 2.3 ... 793
7.27.8 Structure 2.2 Release Notes ... 794
7.27.8.1 Atlassian Marketplace Availability ... 794
7.27.8.2 Other Changes ... 795
7.27.8.3 Supported JIRA Versions ... 795
7.27.8.4 Installation ... 795
7.27.8.5 Upgrading from a Previous Version ... 796
7.27.8.6 Structure 2.2.1 Release Notes ... 796
7.27.9 Structure 2.1 Release Notes ... 797
7.27.9.1 Version Highlights ... 798
7.27.9.2 Changes in Detail ... 798

7.27.9.3 Supported JIRA Versions 799
7.27.9.4 Installation 799
7.27.9.5 Upgrading from a Previous Version 799
7.27.9.6 Detailed Change List 800
7.27.9.7 Structure 2.1.1 Release Notes 800
7.27.9.8 API Changes in Structure 2.1 801
7.27.10 Structure 2.0 Release Notes 801
7.27.10.1 Version Highlights 801
7.27.10.2 Changes in Detail 802
7.27.10.3 Supported JIRA Versions 807
7.27.10.4 Changes for Developers 807
7.27.10.5 Installation 808
7.27.10.6 Upgrading from a Previous Version 808
7.27.10.7 Detailed Change List 809
7.27.10.8 API Changes in Structure 2.0 809
7.27.11 Structure 1.7 Release Notes 810
7.27.11.1 Version Highlights 810
7.27.11.2 Changes in Detail 810
7.27.11.3 Supported JIRA Versions 811
7.27.11.4 Installation 812
7.27.11.5 Upgrading from a Previous Version 812
7.27.11.6 Detailed Change List 812
7.27.11.7 Structure 1.7.1 Release Notes 812
7.27.11.8 API Changes in Structure 1.7 813
7.27.12 Structure 1.6 Release Notes 813
7.27.12.1 Version Highlights 813
7.27.12.2 Changes in Detail 813
7.27.12.3 API Changes 816
7.27.12.4 Supported JIRA Versions 816
7.27.12.5 Installation 817
7.27.12.6 Upgrading from a Previous Version 817
7.27.12.7 Detailed Change List 817
7.27.12.8 API Changes in Structure 1.6 817
7.27.13 Structure 1.5 Release Notes 819
7.27.13.1 Version Highlights 819

7.27.13.2 Changes in Detail 819
7.27.13.3 Supported JIRA Versions 820
7.27.13.4 Installation 820
7.27.13.5 Upgrading from a Previous Version 820
7.27.13.6 Detailed Change List 820
7.27.13.7 Structure 1.5.1 Release Notes 821
7.27.13.8 Structure 1.5.2 Release Notes 821
7.27.13.9 API Changes in Structure 1.5 822
7.27.14 Structure 1.4 Release Notes 823
7.27.14.1 Version Highlights 823
7.27.14.2 Changes in Detail 824
7.27.14.3 Supported JIRA Versions 825
7.27.14.4 API Changes 826
7.27.14.5 Installation 826
7.27.14.6 Upgrading from a Previous Version 826
7.27.14.7 Important Changes to Restoring Structure from Backup 826
7.27.14.8 Detailed Change List 826
7.27.14.9 Structure 1.4.1 Release Notes 826
7.27.14.10 API Changes in Structure 1.4 827
7.27.15 Structure 1.3 Release Notes 828
7.27.15.1 Version Highlights 828
7.27.15.2 Changes in Detail 829
7.27.15.3 Supported JIRA Versions 829
7.27.15.4 Installation 830
7.27.15.5 Upgrading from a Previous Version 830
7.27.15.6 Detailed Change List 830
7.27.15.7 Structure 1.3.1 Release Notes 830
7.27.15.8 Structure 1.3.2 Release Notes 831
7.27.16 Structure 1.2 Release Notes 832
7.27.16.1 Version Highlights 832
7.27.16.2 Changes in Detail 833
7.27.16.3 Supported JIRA Versions 835
7.27.16.4 Installation 835
7.27.16.5 Upgrading from a Previous Version 835
7.27.16.6 Detailed Change List 835

7.27.17 Structure 1.1 Release Notes ... 836
7.27.17.1 1. Version Highlights ... 836
7.27.17.2 2. Changes in Detail ... 836
7.27.17.3 3. Supported JIRA Versions ... 838
7.27.17.4 4. Installation ... 838
7.27.17.5 5. Upgrading from a Previous Version ... 838
7.27.17.6 6. Detailed Change List ... 838
7.27.17.7 Structure 1.1.1 Release Notes ... 838
7.27.18 Structure 1.0 Release Notes ... 839
7.27.18.1 1. Product Highlights ... 839
7.27.18.2 2. Changes from Structure 1.0-rc1 ... 839
7.27.18.3 3. Supported JIRA Versions ... 840
7.27.18.4 4. Installation and Upgrade ... 840
7.27.18.5 5. Detailed Change List ... 841
8 Structure Developer's Guide ... 842
8.1 Structure Developer Documentation ... 842
8.2 Structure Concepts, Developer's Perspective ... 843
8.2.1 Basic Concepts Overview ... 844
8.2.2 A Note on Extensibility ... 844
8.3 Accessing Structure from JIRA Plugin ... 845
8.3.1 Setting Up the Integration ... 845
8.3.1.1 Add dependency to your pom.xml ... 845
8.3.1.2 Import StructureComponents ... 845
8.3.1.3 Have Structure API service injected into your component ... 846
8.3.1.4 Additional Libraries Used in Structure API ... 846
8.3.1.5 Controlling Compatibility ... 847
8.3.1.6 Making Structure Dependency Optional ... 848
8.3.2 Structure Services ... 849
8.3.2.1 Services to Start With ... 850
8.3.2.2 More Power ... 850
8.3.2.3 Extreme Power ... 851
8.3.3 Building Forest Specification ... 851
8.3.4 Reading Structure Content ... 853
8.3.4.1 Figure out Structure ID ... 853

8.3.4.2 Create a ForestSpec ... 853
8.3.4.3 Retrieve ForestSource ... 853
8.3.4.4 Retrieve Forest and its version ... 854
8.3.4.5 Iterate through Forest and get StructureRow instances ... 854
8.3.4.6 Analyze the row and process data ... 854
8.3.5 Changing Structure Content ... 855
8.3.5.1 Forest Coordinates ... 855
8.3.5.2 Applying Forest Action ... 856
8.3.5.3 Inspecting the Results ... 857
8.3.5.4 Effects and Changing Dynamic Structures ... 857
8.3.5.5 Concurrency and Atomicity ... 858
8.3.5.6 Permissions ... 858
8.3.6 Loading Attribute Values ... 858
8.3.6.1 About Attributes ... 858
8.3.6.2 General Approach to Loading Values ... 859
8.3.7 Creating and Adding Folders ... 860
8.3.7.1 Create the Folder entity ... 860
8.3.7.2 Define folder's identity ... 860
8.3.7.3 Add folder to structure ... 861
8.3.8 Creating Dynamic Structures ... 861
8.3.8.1 Create generator instance ... 861
8.3.8.2 Insert generator into the forest ... 861
8.4 Extending Structure Functionality ... 861
8.4.1 Creating a New Column Type ... 862
8.4.1.1 The Plan ... 863
8.4.1.2 The Attributes ... 863
8.4.1.3 AttributeSpec for Status Bar ... 864
8.4.1.4 Status Bar Attribute ... 864
8.4.1.5 Attribute Provider ... 866
8.4.1.6 Client-Side Column ... 867
8.4.1.7 Export Renderers ... 876
8.4.2 Loading Additional Web Resources For Structure Widget ... 879
8.4.2.1 Using Web Resource Contexts ... 879
8.4.3 Declaring a New Generic Item Type ... 880
8.4.3.1 Example ... 880

8.4.3.2 Programmatic Access to Generic Items ... 881
8.4.3.3 Generic Item Permissions ... 882
8.5 Accessing Structure Data Remotely ... 882
8.6 Reference ... 882
8.6.1 Structure Developer Reference ... 882
8.6.2 Structure Java API Reference ... 883
8.6.2.1 Structure API Versions ... 883
8.6.3 Structure Plugin Module Types ... 888
8.6.3.1 structure-attribute-loader-provider ... 889
8.6.3.2 structure-export-renderer-provider ... 889
8.6.3.3 structure-item-type ... 891
8.6.3.4 Generator Modules ... 891
8.6.3.5 structure-effector ... 893
8.6.3.6 structure-effect-provider ... 894
8.6.3.7 structure-query-constraint ... 895
8.6.3.8 new-structure-template ... 895
8.6.4 Structure REST API Reference ... 896
8.6.4.1 General Notes ... 896
8.6.4.2 REST Resources ... 897
8.6.4.3 Structure Resource ... 897
8.6.4.4 Forest Resource ... 917
8.6.4.5 Item Resource ... 920
8.6.4.6 Value Resource ... 924
8.6.4.7 Attribute Subscription Resource ... 928
8.6.4.8 Configuration Resource ... 933
8.6.5 Structure JavaScript API Reference ... 944
8.6.5.1 JavaScript API Functions ... 944
8.6.5.2 JavaScript API Classes ... 946
8.6.5.3 JavaScript API Objects ... 962
8.6.6 Web Resource Contexts ... 965
8.7 API Usage Samples ... 965
8.7.1 Download ... 965
8.7.2 Example List ... 966
8.8 Archived Documentation ... 966
8.8.1 Structure 3 API Changes ... 967

8.8.1.1 State of the API ... 967
8.8.1.2 Conceptual Changes ... 967
8.8.1.3 REST API ... 969
8.8.1.4 Java API ... 971
8.8.2 Creating a New Synchronizer ... 973
8.8.2.1 Implement StructureSynchronizer ... 974
8.8.2.2 Define structure-synchronizer Module ... 974
8.8.2.3 Test Thoroughly ... 974
8.8.2.4 Sample Project ... 974
8.8.3 structure-synchronizer ... 974
8.8.3.1 Module description sample ... 974
8.9 Open Source Licenses ... 975
9 Download ... 978
9.1 Download Structure Plugin ... 978
9.2 Download Structure Extensions ... 978
9.3 What's Next? ... 979
9.4 Documentation ... 979
9.5 Download Archive ... 979
10 Structure Roadmap ... 998
10.1 Versions and Dates ... 998
10.2 Roadmap ... 998
11 Send Feedback ... 1000
11.1 Please tell us what you think! ... 1000

1 Structure – Project Management at Scale

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

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

Structure has a number of powerful features:

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

Structure also comes with Structure.Testy, our extension for testing checklists.

- Getting Started with Structure(see page 49)
- Structure User's Guide(see page 73)
- How-To Articles(see page 423)
- Structure Administrator's Guide(see page 458)
- Frequently Asked Questions(see page 561)
- Release Notes(see page 572)
- Structure Developer's Guide(see page 842)
- Download(see page 978)
- Structure Roadmap(see page 998)
- Send Feedback(see page 1000)

Sandbox / Demo Server[1]

If login and password are requested, use demo/demo.

Atlassian Marketplace Listing[2]

Get more information about pricing and check out user reviews.

Download(see page 978)

You can install the Structure app via Add-on Manager in Jira. If there is any trouble with downloads, you can always get it here(see page 978).

Compatibility

Jira	7.13 and later (all server editions and Jira Data Center are supported)
Confluence	6.13 and later (supported via Structure.Pages)

⚠ The articles and resources on this page are intended for Structure Server and Data Center users. If you are using our cloud version, see Structure Cloud[3].

1 https://demo-structure3.almworks.com/secure/StructureBoard.jspa?s=187&os_username=demo&os_password=demo
2 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
3 https://wiki.almworks.com/display/strcloud/Structure+Cloud

2 Getting Started with Structure

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.

2.1 Before You Begin

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

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

And one last note before we begin. This guide is only intended to cover the essential information for getting started and working with Structure. For an in-depth discussion of the many capabilities and features Structure has to offer, please refer to our Structure User's Guide(see page 73).

That's it! Now let's get started: Creating Your First Structure(see page 49)

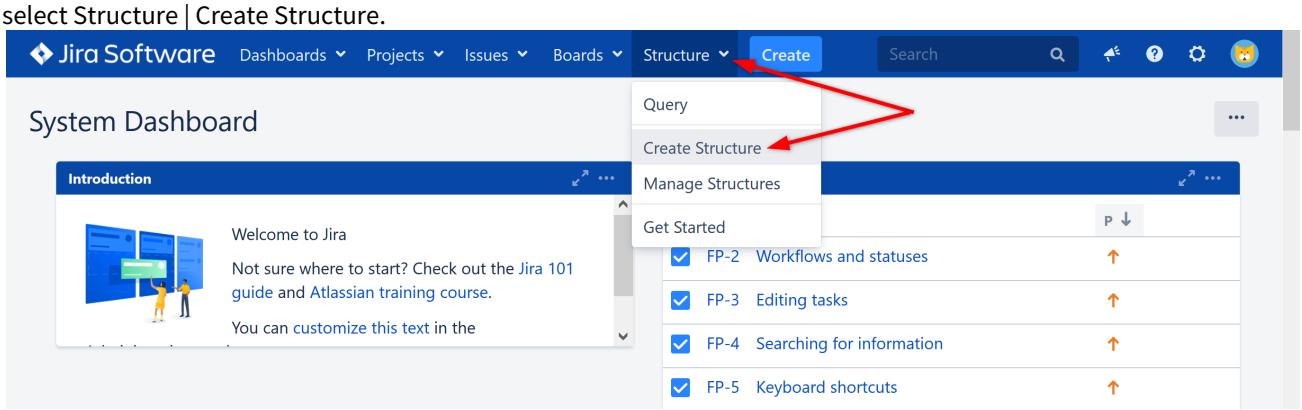
2.2 Creating Your First Structure

While there are endless ways to build a structure, we're going to show you some of our most popular (and useful) approaches, including our powerful automation tools(see page 50) and steps for adding issues manually(see page 60).

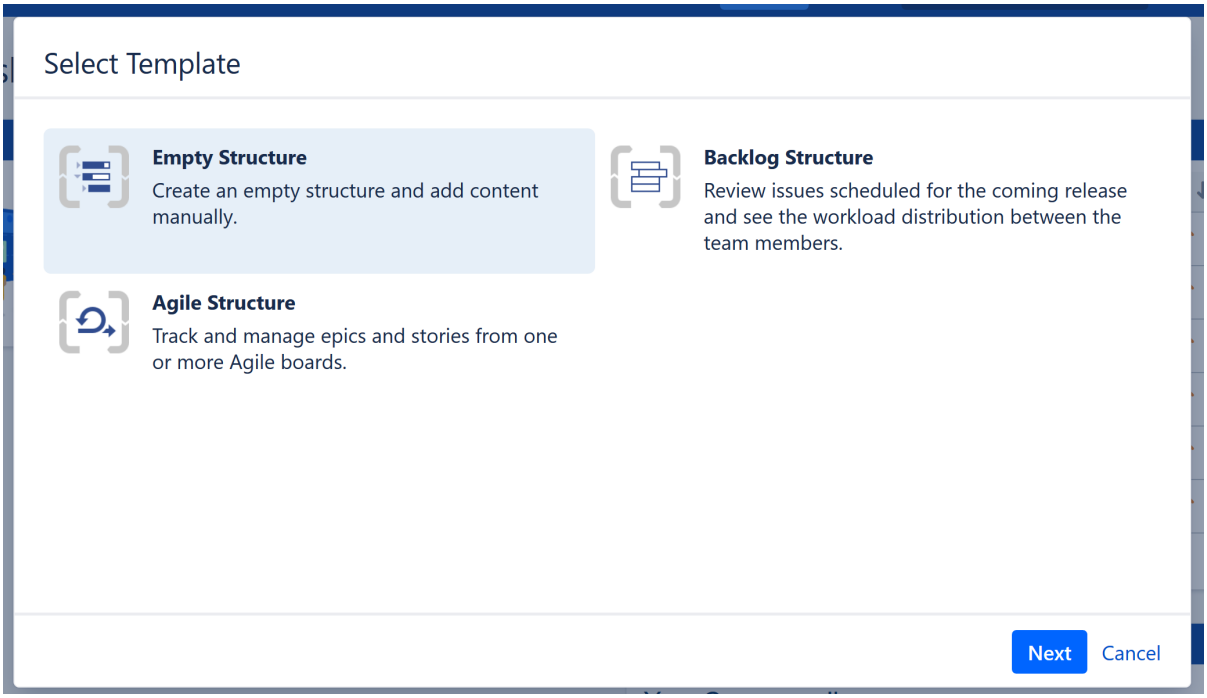
We encourage you to experiment with the tools presented here, make several new structures and find some templates that will work well for you and your team

2.2.1 Your First Structure

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



You 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(see page 362)). When you're finished, click **Create**.

2.2.2 Next Steps

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

- Building a Structure with Automation(see page 50)
- Building a Structure Manually(see page 60)

2.2.3 Building a Structure with Automation

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

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

2.2.3.1 A (Very) Brief Overview of Automation

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

Automation uses **generators** - special rules that tell Structure what issues to show you from Jira and where to place them within your structure. Every time you open Structure, these generators will run again and completely rebuild your structure, based on the current information available. In this way, you know that your structure is always up-to-date and relevant. To learn more about how Automation works and the types of generators available, see Automation(see page 122).

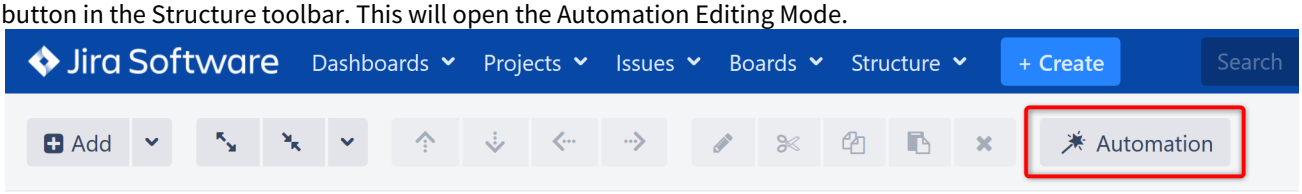
For now, we'd like to show you you how to build two of our most popular (and useful) automated structures, both of which can be built in just a few minutes.

- Top Down Automation to Manage Issues Across Multiple Projects(see page 51)
- Bottom Up Automation for Backlog Grooming(see page 56)

2.2.3.2 Top Down Automation to Manage Issues Across Multiple Projects

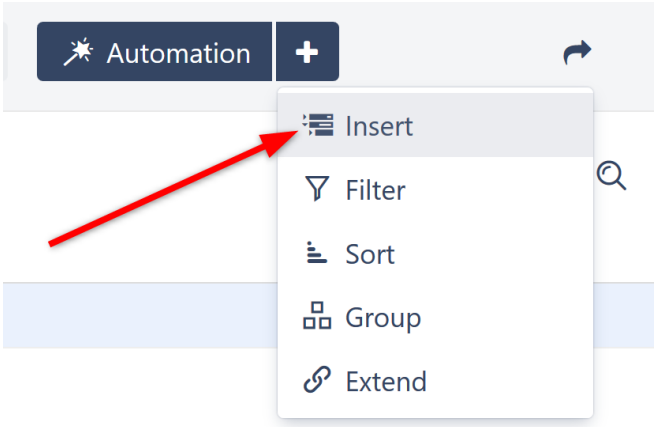
One of the great advantages to using Structure is the ability to manage issues from multiple projects in a single location. The following guide will walk you through one approach for adding Epics, Stories, and Tasks from multiple projects into a single structure.

Starting with a brand new, blank structure (see Creating Your First Structure(see page 49)), click the **Automation** button in the Structure toolbar. This will open the Automation Editing Mode.



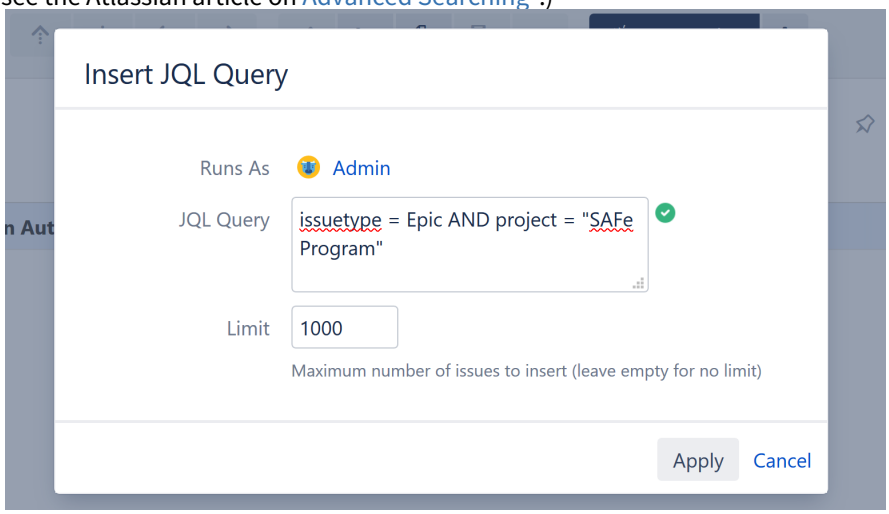
Step 1: Insert Epics

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



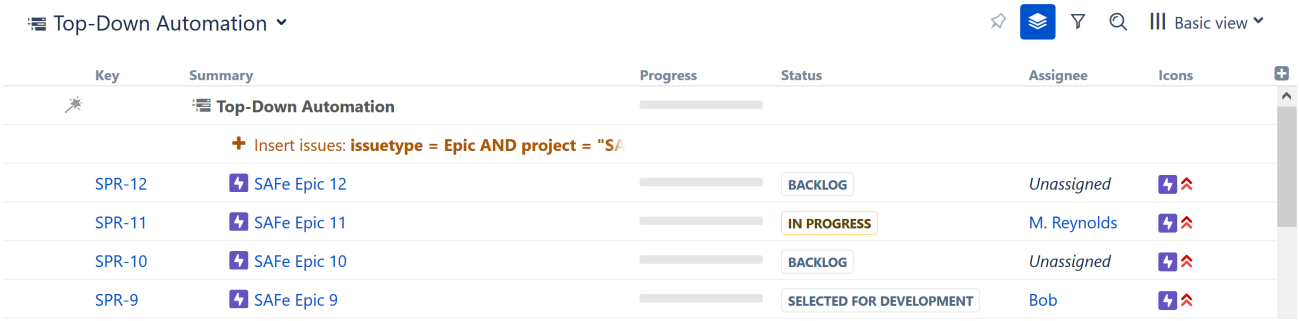
You can import items from an Agile board or existing structure, or you can use a JQL query or text search. Since we want to view issues from across multiple projects, let's use a simple JQL Query: "Issuetype = Epic". If you want to narrow your results to specific projects, you can specify that within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

see the Atlassian article on Advanced Searching[4].)



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

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



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

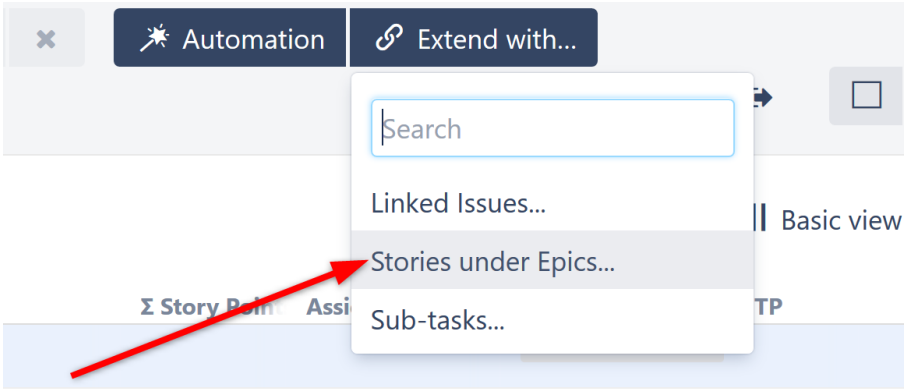
- To make changes to a generator, double-click it.
- To delete a generator, highlight its row and click the delete icon (**x**) or delete key.

Step 2: Extend with Stories

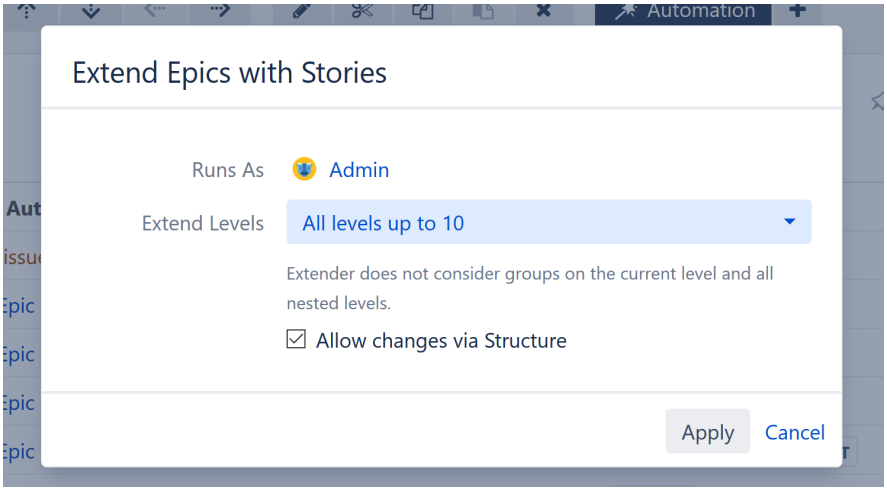
Now it's time to **Extend** our results to include Stories and Sub-tasks. To do this, make sure the top line of your structure, containing the structure's name, is still highlighted (this makes sure we are applying the generator to the entire structure) and click the Add Generator button **(+)** again.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

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

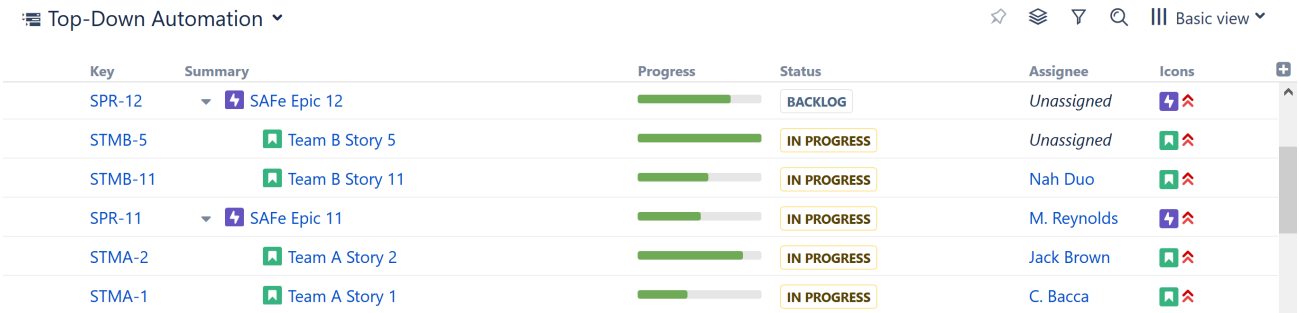
4 https://confluence.atlassian.com/jiracoreserver073/advanced-searching-861257209.html



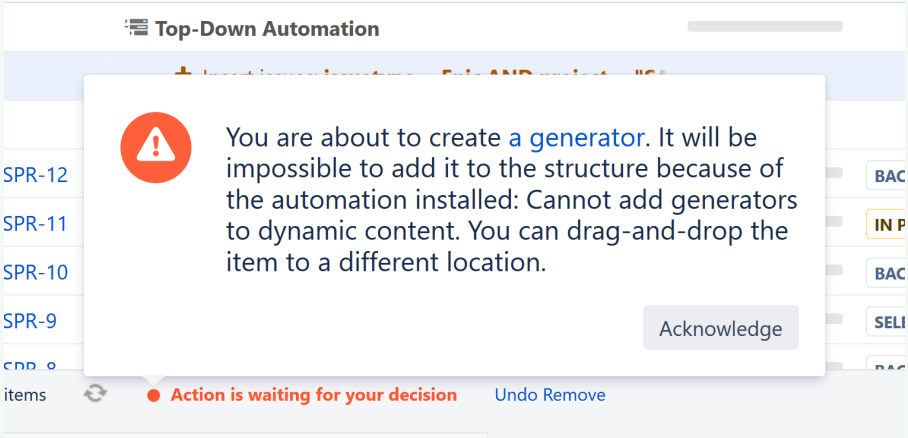
The Extend generator gives you the option to "Allow changes via Structure." When selected, moving a story from one epic to another within Structure will also move the story within Jira. This option is selected by default.



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



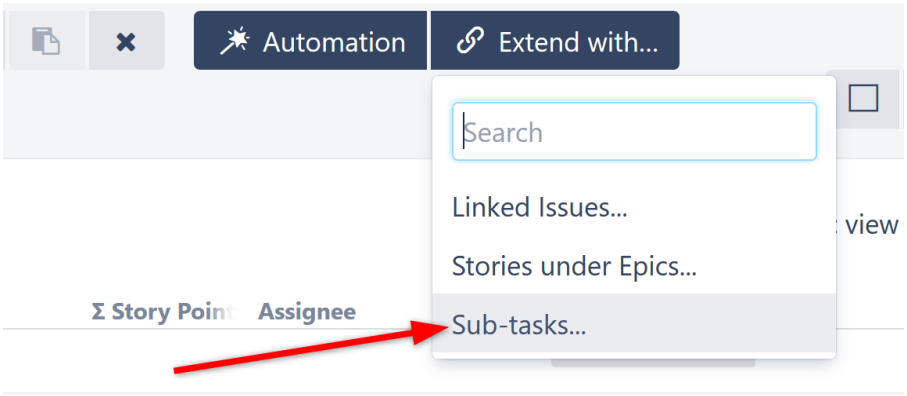
If you forget to highlight the top line of your structure before adding the Extend generator, you may get the following error:



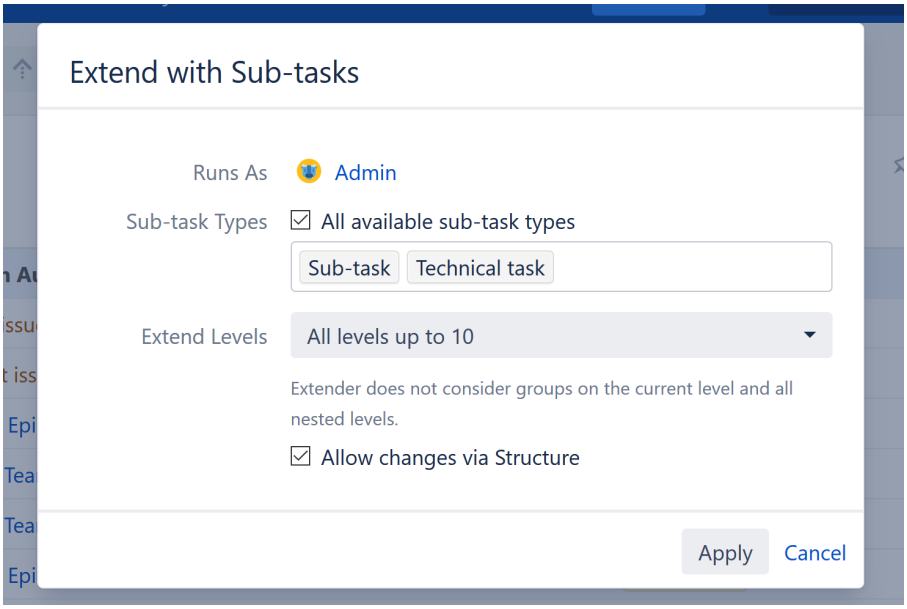
Since we started with a global generator (affecting the entire structure), all other generators need to be global as well, to avoid any conflicts. Simply highlight the top line of your structure (where it says the structure's name), and try again.

Step 3: Extend with Sub-tasks

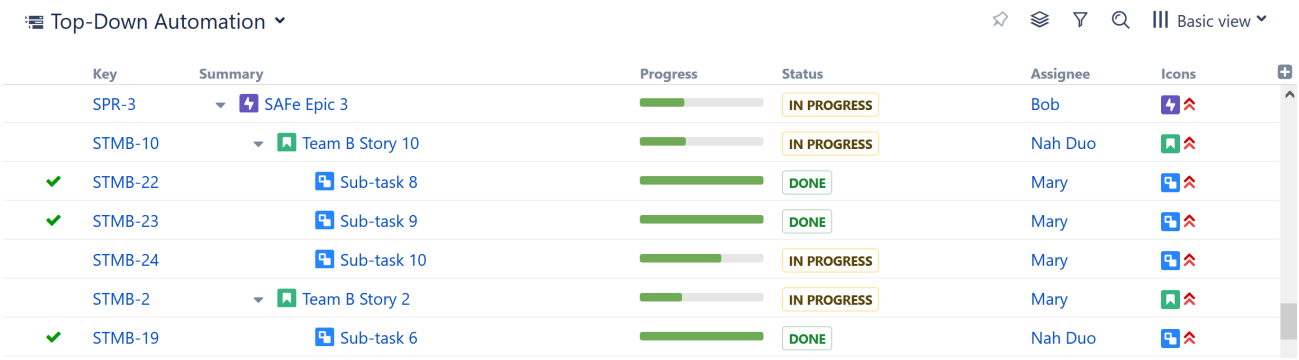
Finally, we'll add Sub-tasks under Issues. Once again, make sure the top line of your structure is still selected (highlighted), click the '**+**' next to the Automation button and select **Extend**. This time, select **Sub-tasks**.



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



Your issues should now appear in a hierarchy, based on the Automation rules we selected. If you want to re-arrange items or move a sub-task from one story to another (or copy it from one to another), you can do so from within your structure. (We'll cover that in Working with Structure(see page 63).)



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(see page 122) section of our Structure User's Guide.

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

Next Steps

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

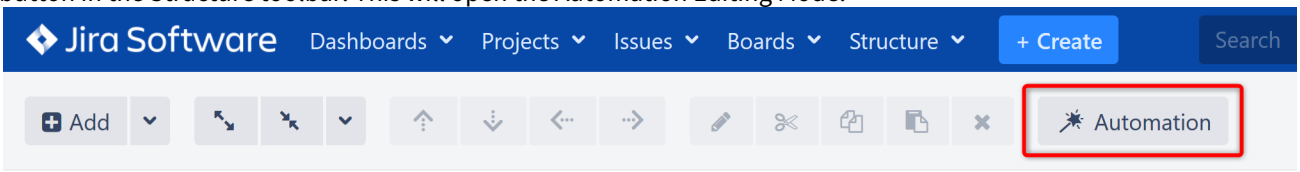
Bottom Up Automation for Backlog Grooming(see page 56)

2.2.3.3 Bottom Up Automation for Backlog Grooming

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

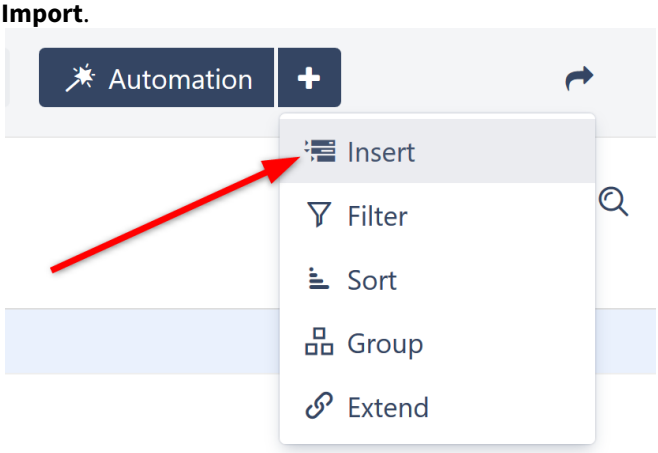
In the last tutorial, we started at the top of our hierarchy (Epics) and worked our way down. This time, we're going to start at the bottom and insert all of our active stories.

Starting with a brand new, blank structure (see Creating Your First Structure(see page 49)), click the **Automation** button in the Structure toolbar. This will open the Automation Editing Mode.



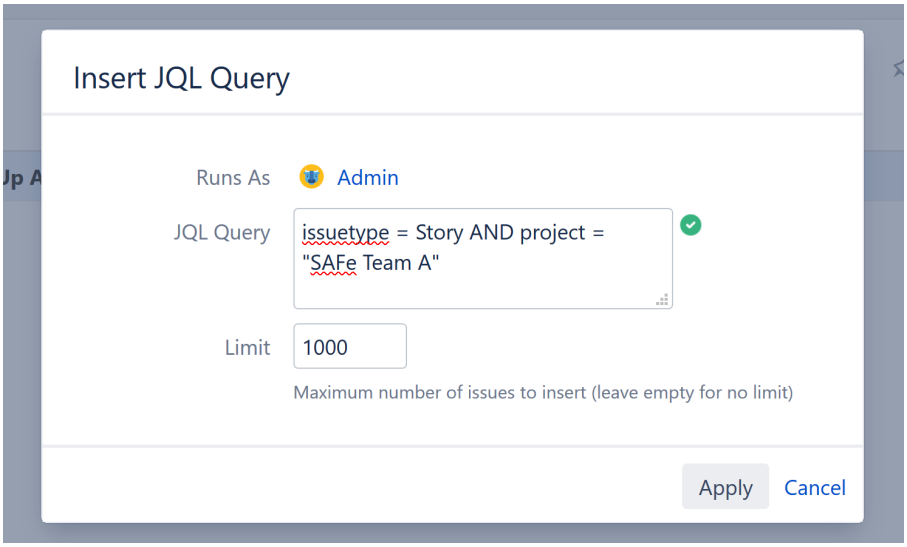
Step 1: Insert Stories

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

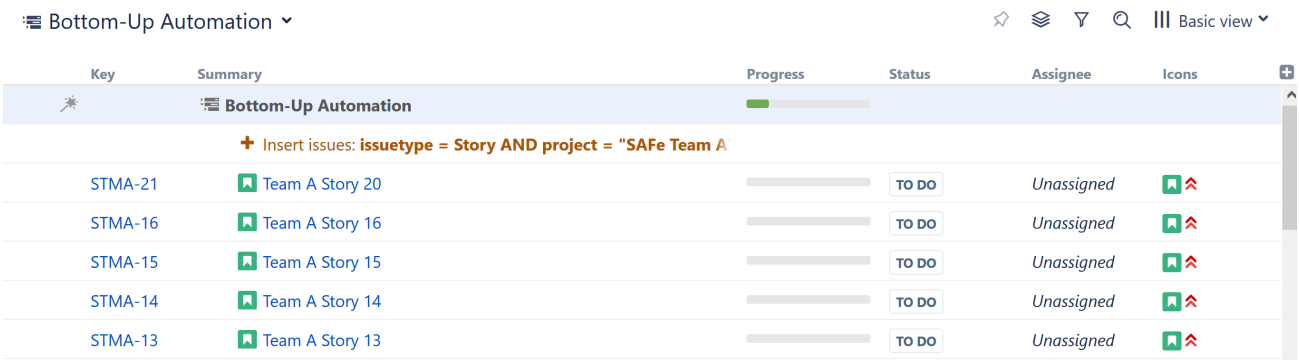


This time, we'll use the JQL Query: "Issuetype = Story". Just as with our top-down structure, you can also narrow your results by specifying projects (see Advanced Searching[5] for more information about using JQL).

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



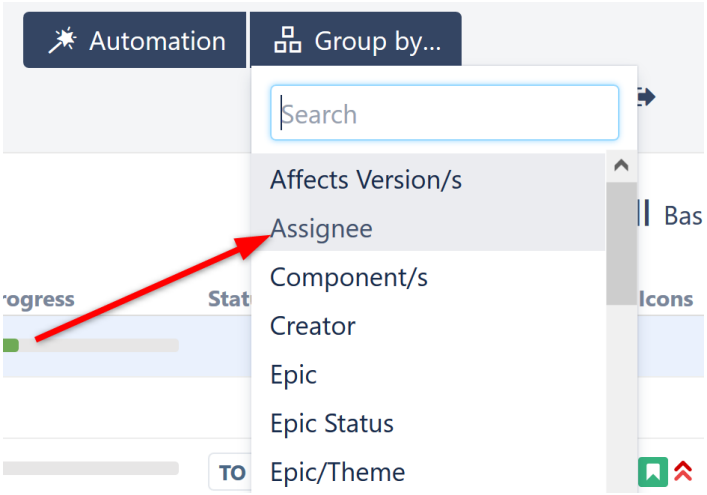
Once again, the Insert generator will pull all issues that match our query into the new structure. In this case, they will add the stories from our specified project.



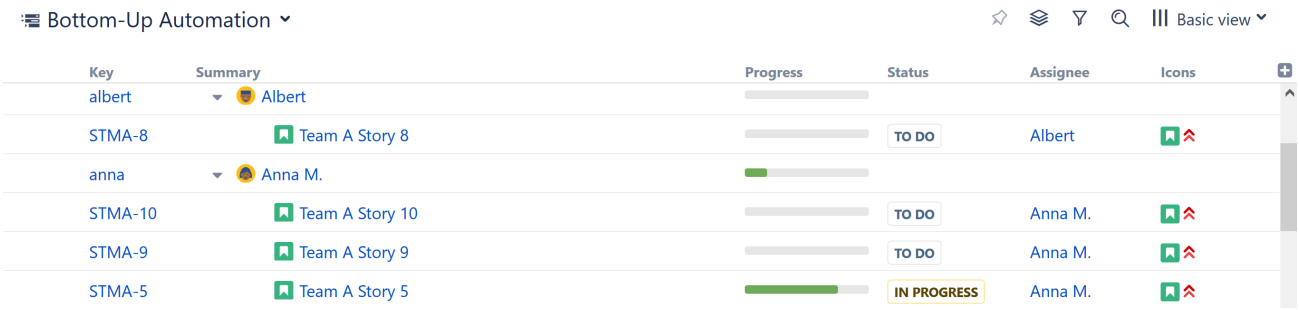
These will serve as the bottom-most level of our hierarchy, and we will use the Group generator to add levels above this.

Step 2: Group by Assignee

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



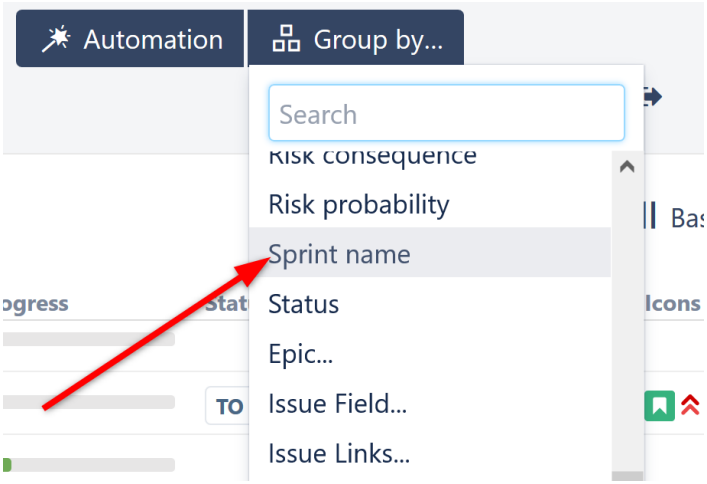
At this point, all of your stories should appear beneath their respective assignee.



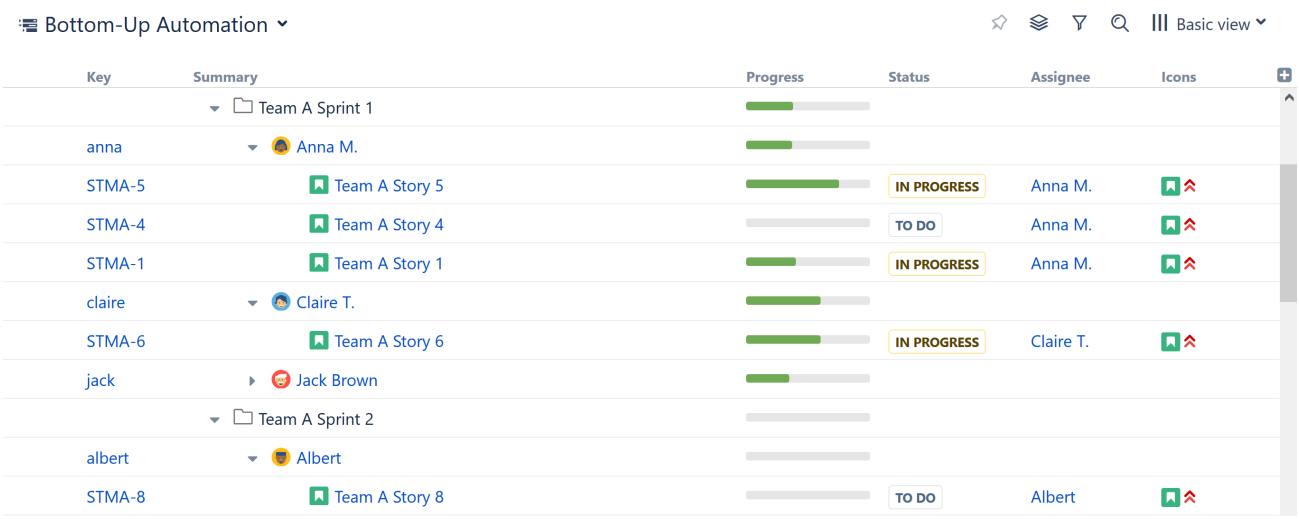
Now we'll add another level to our hierarchy, so we can view the work being done for each sprint.

Step 3: Group by Sprint

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



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



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

Taking it Further

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

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

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

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

> ⚠ These are just examples of a few ways you can build structures using Automation. There are hundreds of other possibilities, allowing you to create the perfect hierarchy for your business needs – or several perfect hierarchies! In this guide, we focused on building down with extenders and building up with groupers, but you don't have to work in just one direction. You can start in the middle and add both extenders and groupers. If you really want to customize things, you can add multiple inserters, group different levels by different attributes, sort specific levels, and more.
>
> To learn more about some of these advanced steps, check out our full articles on Automation(see page 122).

Next Steps

If you're interested in learning about adding items to a structure manually, continue on to Building a Structure Manually(see page 60)

If you would rather keep things simple, jump ahead to Working with Structure(see page 63)

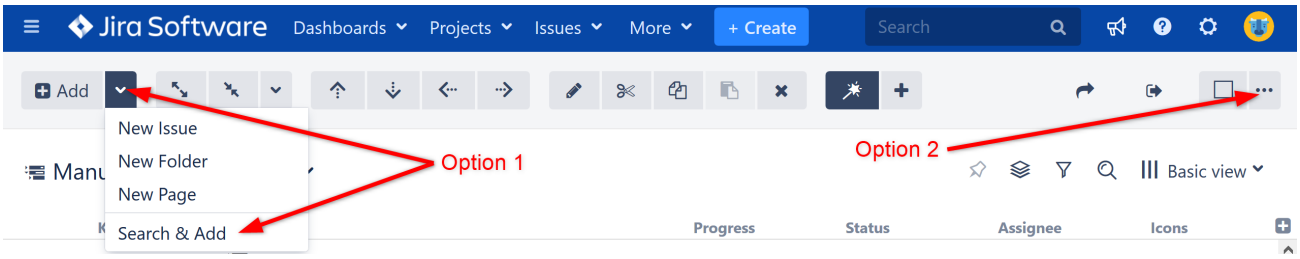
2.2.4 Building a Structure Manually

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.

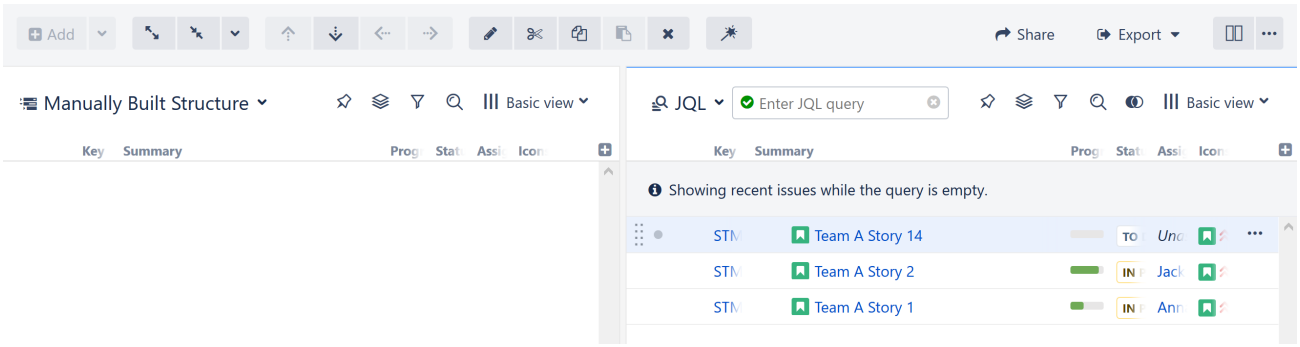
> Changes to issue properties from within Structure are reflected in Jira, regardless of how your structure is built.

2.2.4.1 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**.



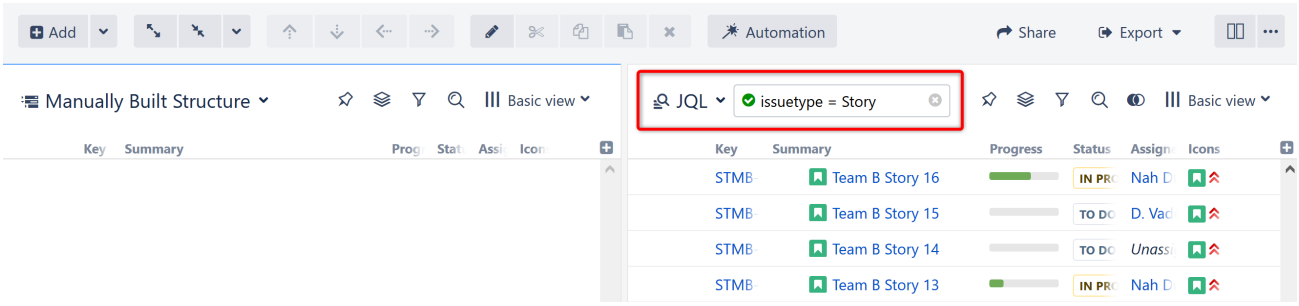
Either of these methods will open a second panel to the right of your structure. In this panel, you can enter a JQL query to search for issues you may want to add to your structure.



2.2.4.2 Step 2: Search for Issues

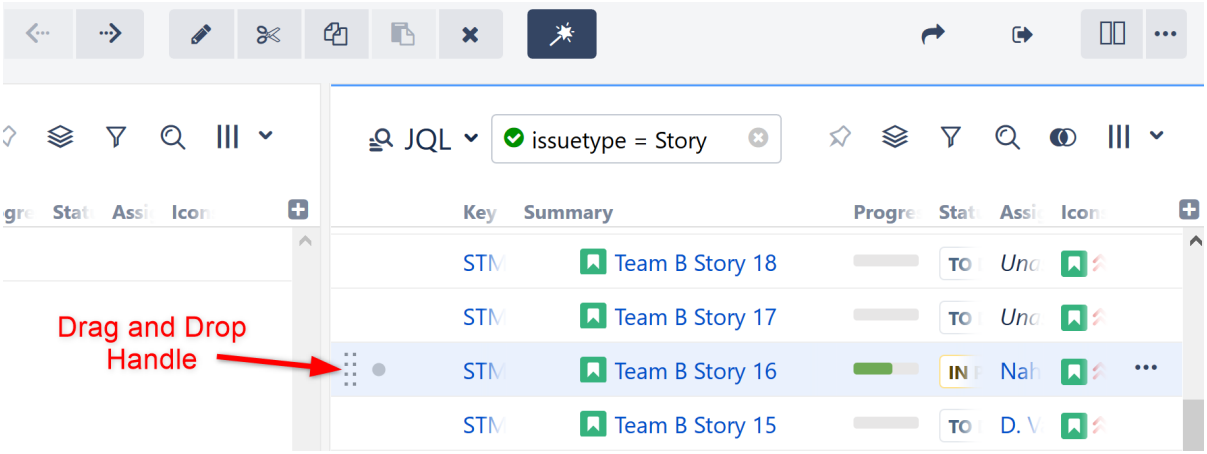
To find issues, enter a JQL query(see page 180) into the search box. (You can also search by plain text or add items from existing structures, but those are outside the scope of this tutorial – for more information, see Adding Issues(see page 94)).

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

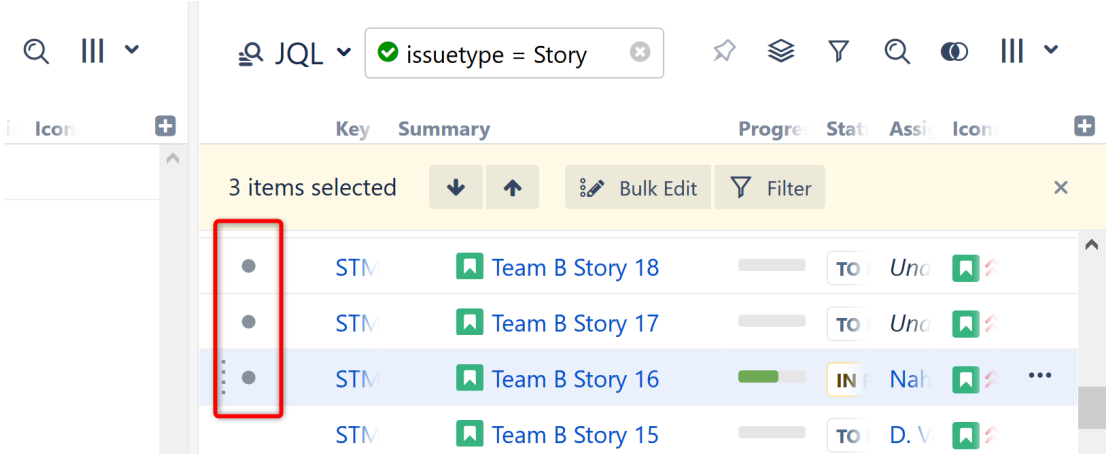


2.2.4.3 Step 3: Add Issues to Your Structure

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.



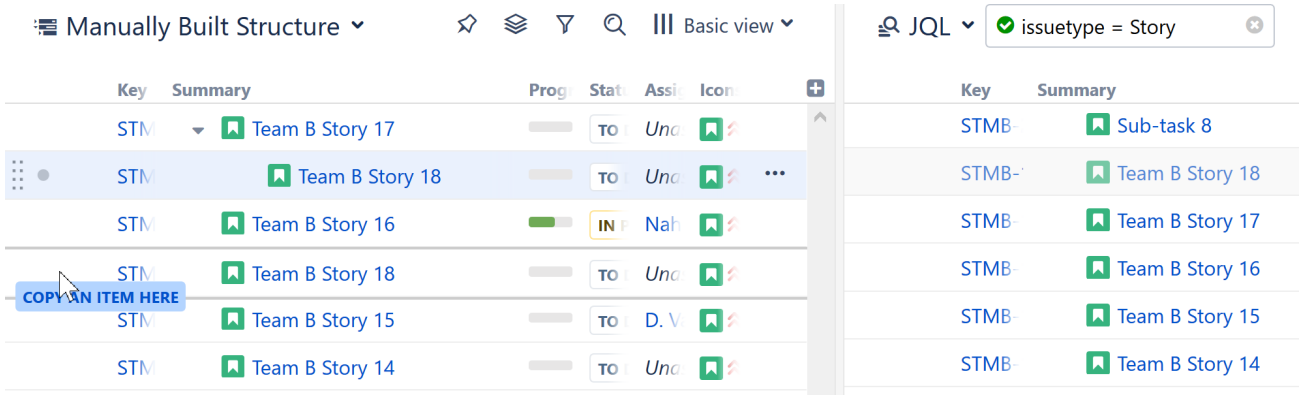
To add more than one issue at a time, click the small circle next to the drag and drop handle for each issue you want to add. Once you have selected all of the issues you want, use the drag and drop handle for any of your selected issues – this will allow you to drag all of them into your structure.



Adding the Same Issue Multiple Times

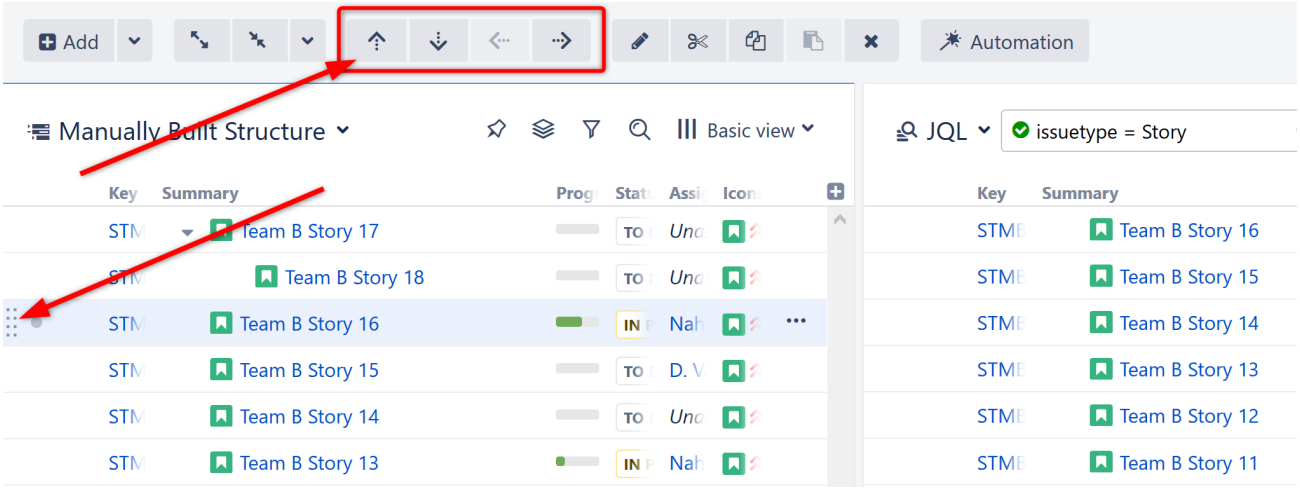
Issues can appear more than once within a structure. For example, a single task might be a requirement for multiple stories - so you may want to include it beneath all of those stories.

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).



2.2.4.4 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(see page 105) or the control buttons in the toolbar(see page 76) above the structure board.



We'll discuss more about moving issues in Working with Structure(see page 63).

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.

2.2.4.5 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(see page 63)

2.3 Working with Structure

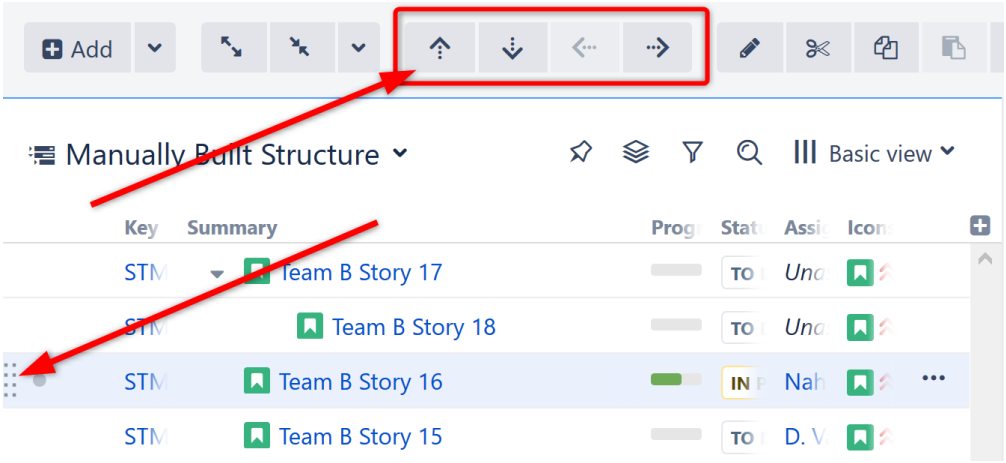
2.3.1 Moving, Adding and Removing Items from a Structure

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

2.3.1.1 Moving Items

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

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



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

2.3.1.2 Adding Items

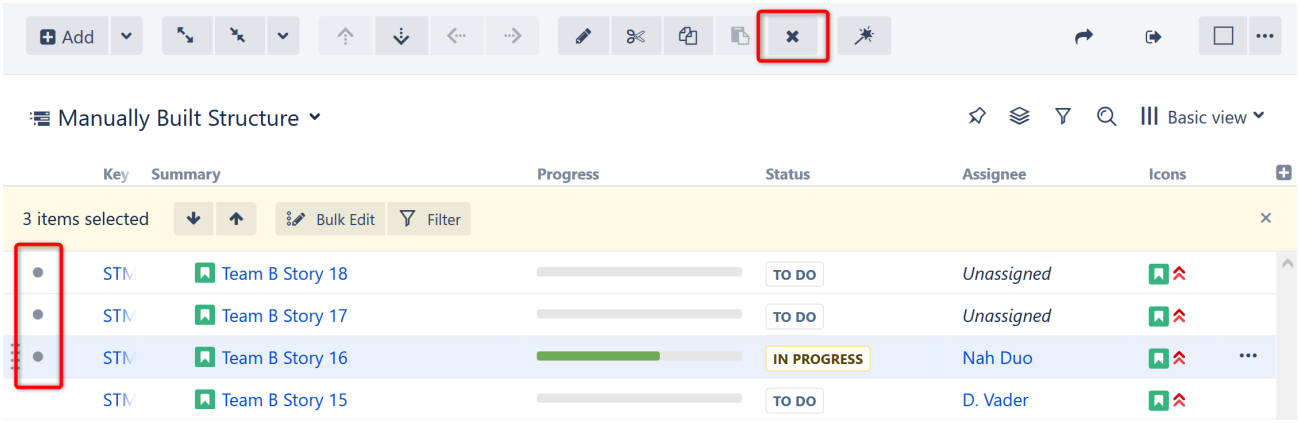
To add items (issues, folders, etc.) to a structure:

- Use the Add menu in the Structure Toolbar,
- Drag and drop the item from Jira or another structure (see Building a Structure Manually(see page 60)), or
- Use Automation (see Building a Structure with Automation(see page 50)).

2.3.1.3 Deleting Items

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

To delete multiple items, select each item by clicking the gray dot at the beginning of each item row (see Selecting Multiple Items(see page 101)). Once you have selected all the items you want to remove, click the delete key in the toolbar or use your keyboard.

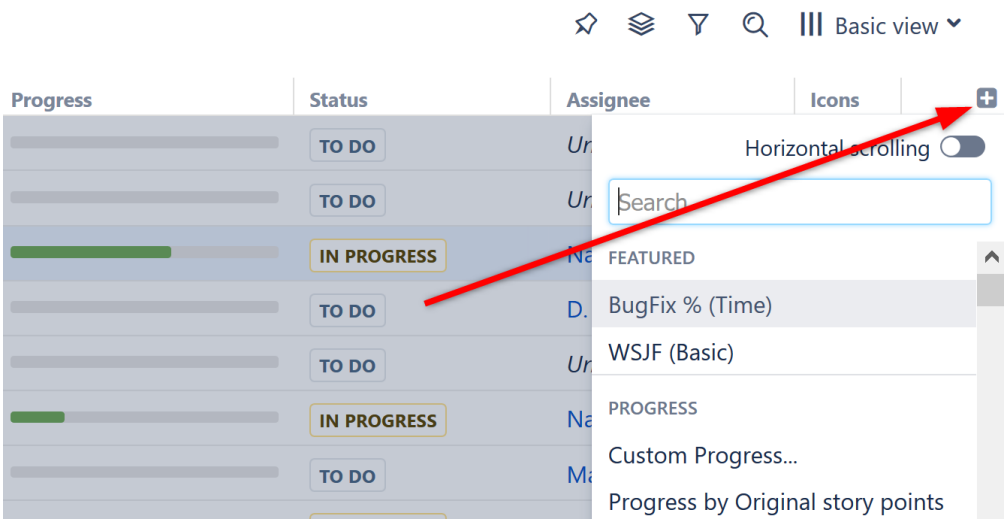


Removing an issue from a structure does not delete the issue itself. It just removes it from the current structure.

2.3.2 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.



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

2.3.2.1 Special Columns and Views

Structure also offers several very useful custom columns, including:

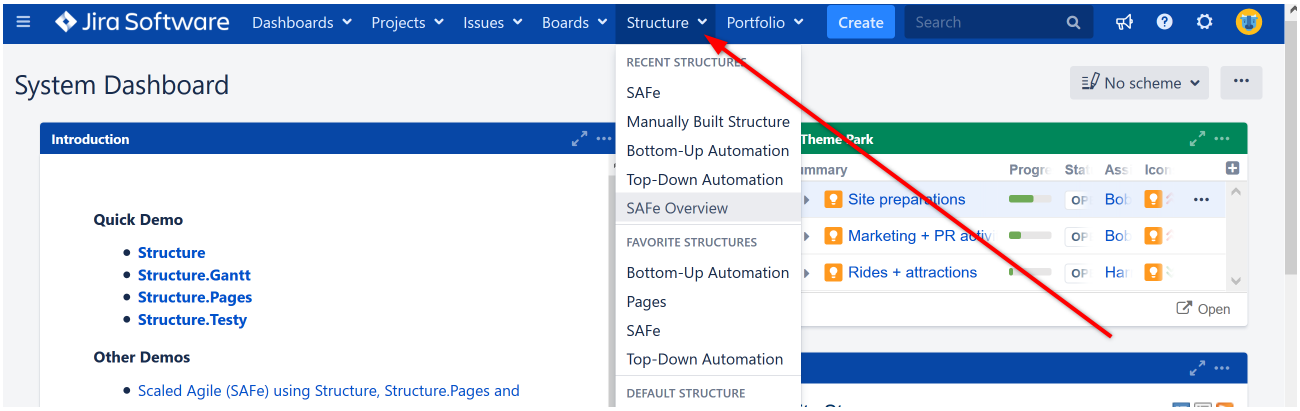
- Progress(see page 300) - Progress can be calculated based on the **Time Tracking**, **Status** or a custom field such as **Story Points**. To review (or change) your Progress calculation settings, hover over the Progress title in the column header and click the small triangle. You can find all the details on how the progress is calculated in the full documentation(see page 300).
- Work Logged(see page 315) - The Work Logged column displays the work logged for every issue over a selected period.
- Σ **Totals**(see page 297) - Structure offers a number of custom columns - including **Totals** columns. These special columns display the value for the selected field as a sum of the values of its children. This is especially useful for showing such things as **Estimates**, **Time Logged**, and **Story Points**.
- Formulas(see page 63) - Formula columns allow you to calculate values based on issue fields and other attributes using custom formulas. Structure offers several built-in formulas, or you can create your own.

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

2.3.3 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.

2.3.4 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(see page 83).

2.3.5 Next Step

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

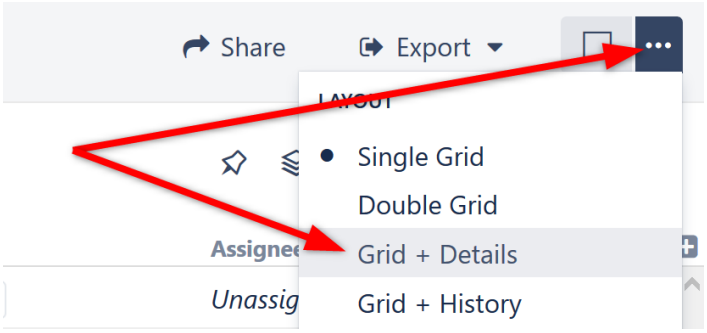
Working with Issues in Structure(see page 66)

2.3.6 Working with Issues in Structure

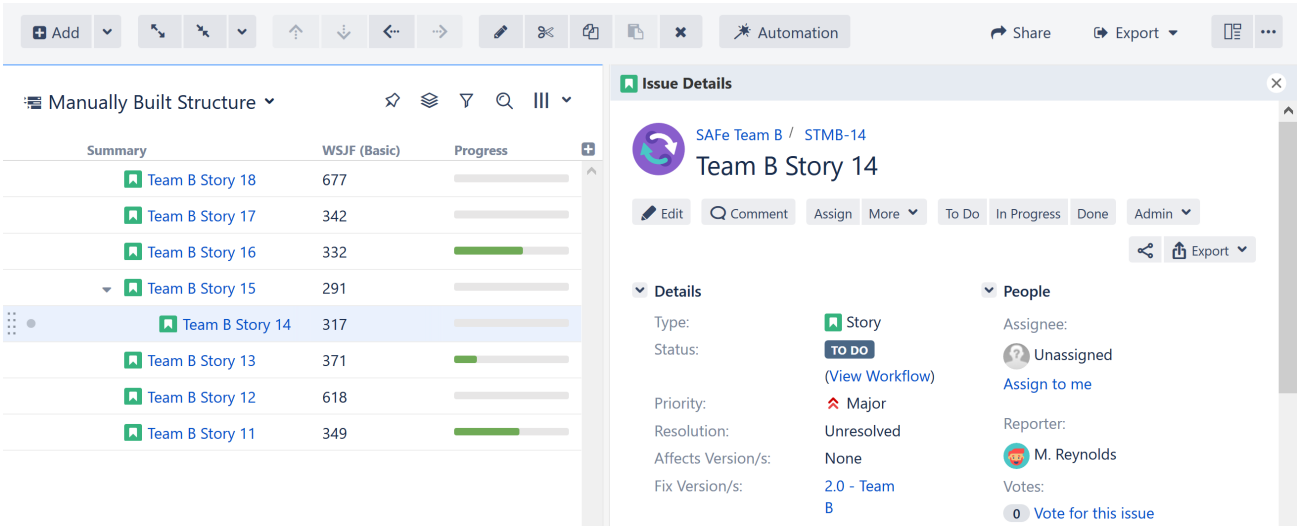
You can view issue details and make changes to issue properties directly from Structure.

2.3.6.1 Issue Details Panel

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



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

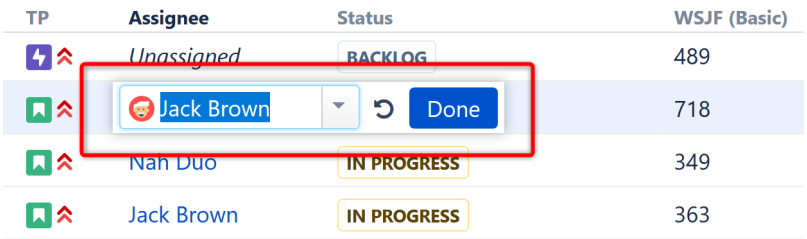


2.3.6.2 Editing Issues

There are multiple ways to edit issues directly from Structure.

Issue Columns

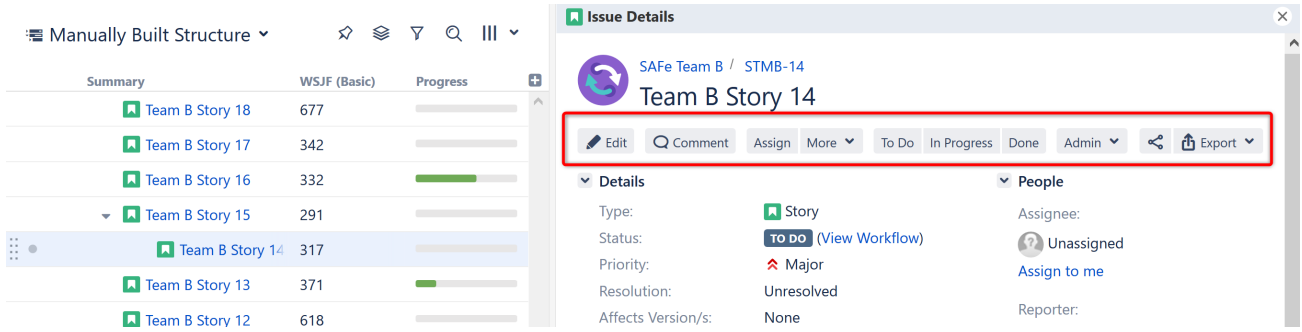
The simplest way to edit an issue is to make changes directly within your structure. If the field you want to change is visible from within the structure, simply double-click the current value to enter editing mode(see page 114).



If the field is not visible, you can add the column(see page 331) 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[6], view and add comments[7], share[8], view history[9], view development information[10], and more.



For specific information on working with and editing issues, please refer to the Jira documentation[11].

Jira Actions Column

Using the Jira Actions Column (far right column of the structure board), you can conduct many of the same actions available within the Issue Details Panel directly from you structure. This column works like the similar column on the Jira Issue Navigator page and lets you log work, apply workflow actions and use other Jira actions(see page 121) available for the issue.

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

2.3.6.3 Next Steps

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

Search Filter and Sort(see page 69)

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

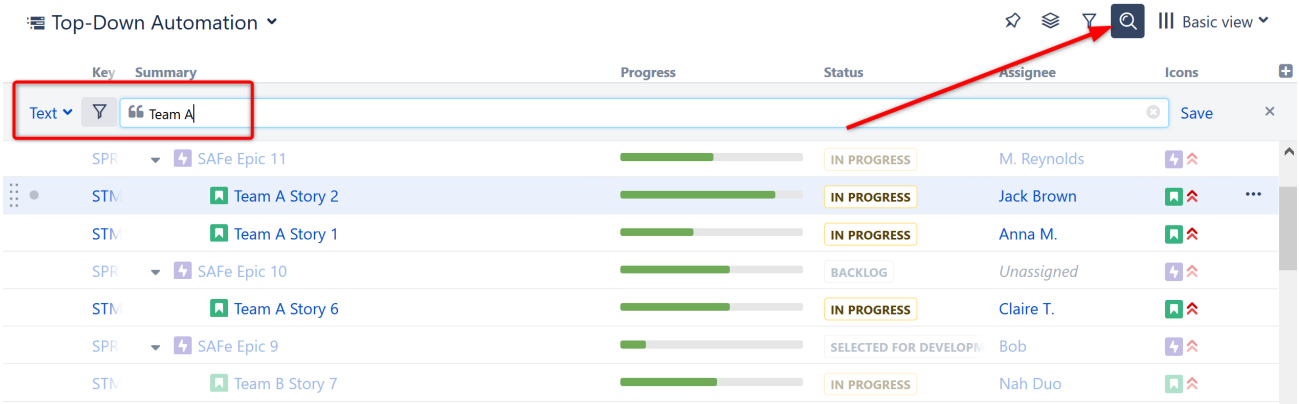
2.3.7 Search Filter and Sort

Once you add issues from several projects, you will likely want a way to organize, search and filter them, so you can focus in on specific information needs. Some of that can be handled using Automation(see page 122), which we introduced in Building a Structure with Automation(see page 50). But those operations affect the look and content of a structure for all users. What if you want to reorganize a structure temporarily? Or quickly hide issues you don't need to see at that precise moment?

This is where search and transformation come in.

2.3.7.1 Search

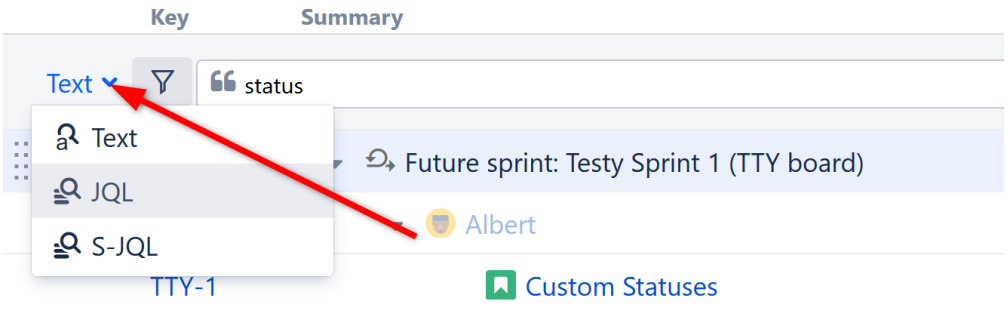
The Search function allows you to highlight issues within your structure. Click the **Search** button on the Structure Panel Toolbar and enter your query into the Search panel.



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

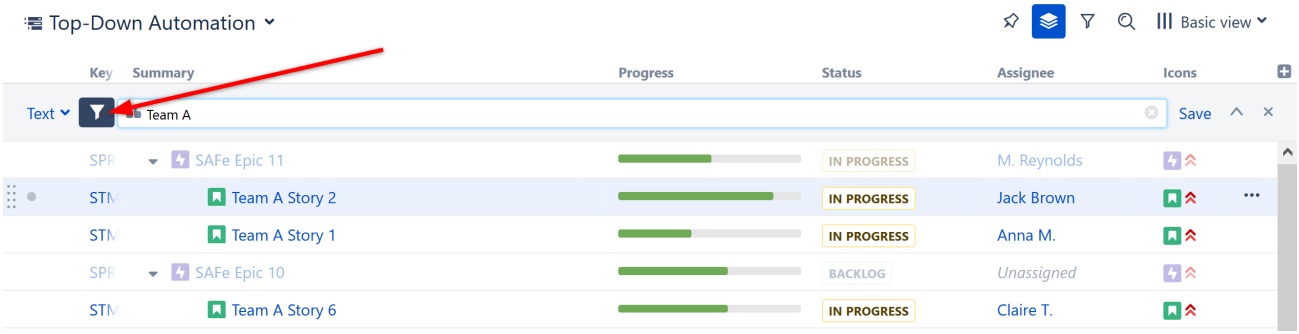
Search Options

You can search for issues(see page 180) in the current structure using a text search, JQL or Structure's own query language, S-JQL. To switch between these modes, click the name of the currently selected mode and select the one you need from the menu.



2.3.7.2 Filter

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



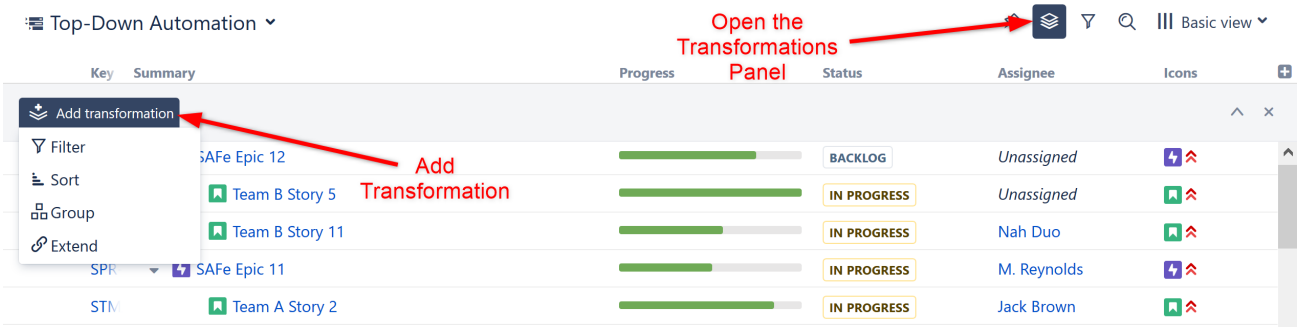
2.3.7.3 Transformations

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

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

To apply a Transformation:

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



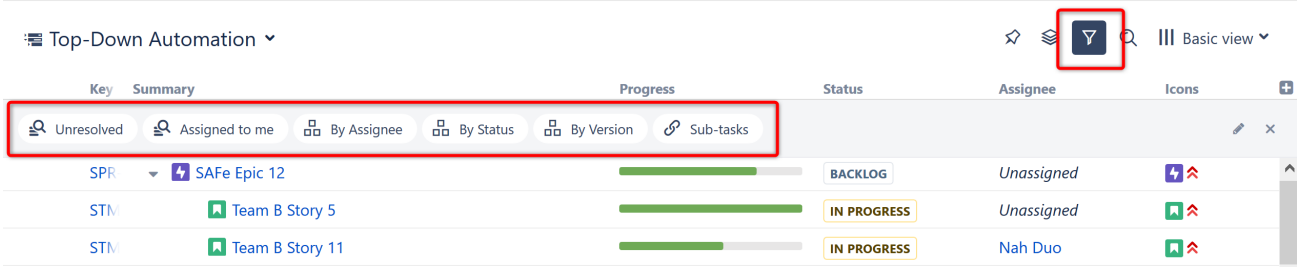
When you are finished with the transformation, simply click the **x** on the right side of the Transformations Panel and the transformation will be removed.

For more information, see Transformations(see page 184).

2.3.7.4 Quick Transformations

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

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



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

To activate a quick transformation

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

To deactivate a quick transformation

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

To learn more about Quick Transformations and how to create your own, see Quick Transformations(see page 188).

2.3.7.5 Next Steps

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

Help and Support(see page 71)

2.4 Help and Support

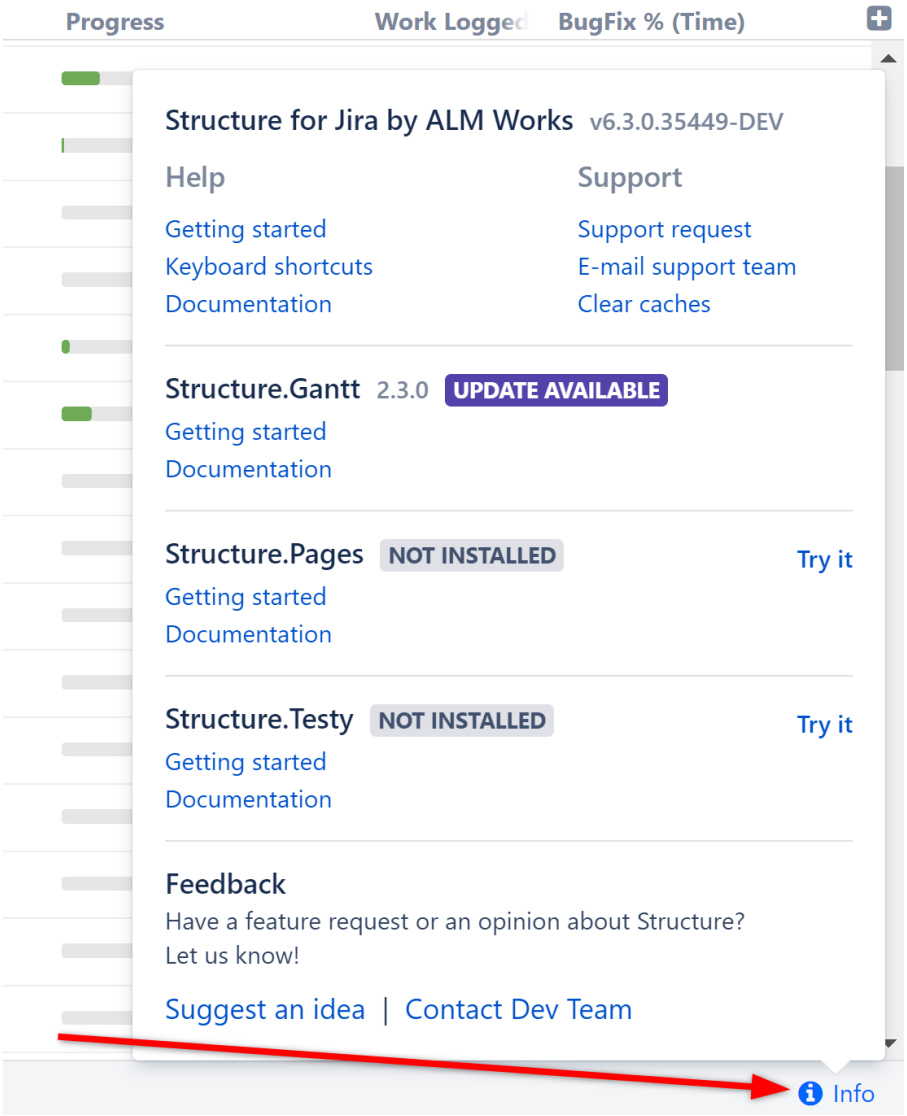
You have now seen how easy it is to create and work with structures – and hopefully you're excited to try creating some new structures all on your own. Good luck and have fun!

2.4.1 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(see page 73) or contact our support team.

2.4.2 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 documentation and keyboard shortcuts, version and update information for Structure and Structure extensions, and links for our support services.



3 Structure User's Guide

If you're new to Structure, we suggest you begin by reviewing our Getting Started Guide.(see page 73)

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

Search

Expand all(see page 73) Collapse all(see page 73)

3.1 Navigating Structure

- Basic Concepts(see page 73)
- Structure Menu(see page 74)
- Main Structure Toolbar(see page 76)
- Structure Widget Overview(see page 78)
    - Navigating Between Items(see page 79)
    - Switching Between Structures(see page 80)
    - Using Structure Widget for Searching(see page 81)
- Jira Pages with Structure(see page 83)
    - Structure Board(see page 83)
        - Making Structure Board Your Jira Home(see page 85)
    - Structure on the Issue Page(see page 85)
        - Structure Options for the Issue Page(see page 88)
    - Structure on the Project Page(see page 90)
    - Structure on Agile Boards(see page 91)
- Sharing a Perspective(see page 92)

3.1.1 Basic Concepts

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

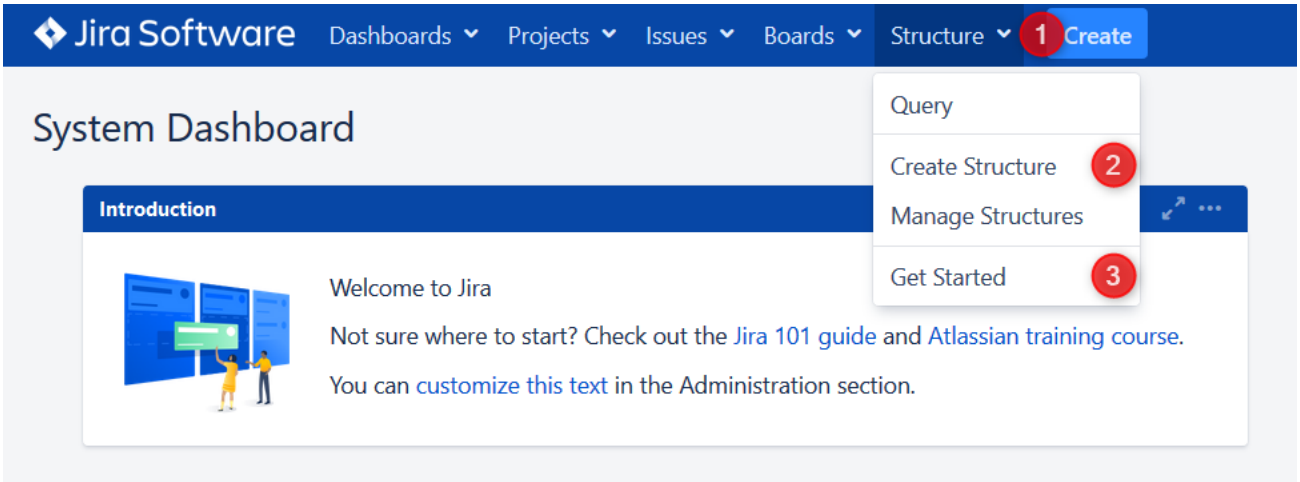
Structure (vs. structure)	*Structure* is the name of our product. In our documentation we differentiate between ***Structure***, the app, and the ***structures*** that you build with it using capitalization. When you see "*Structure*" with a capital "*S*" we are referring to the app. When you see "*structure*" with a lowercase "*s*" we are referring to the the structures you create in the app.
structures contain Jira issues	Think of a structure as a container that may be filled with Jira ***issues*** from a single, or multiple Jira projects. Within this container you may organize the issues into arbitrary groups of hierarchical lists. For example, you may wish to group your issues by type, by assignee, or by priority—or by some combination thereof. In fact, you may organize the issues in your structure any way you'd like.

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

3.1.2 Structure Menu

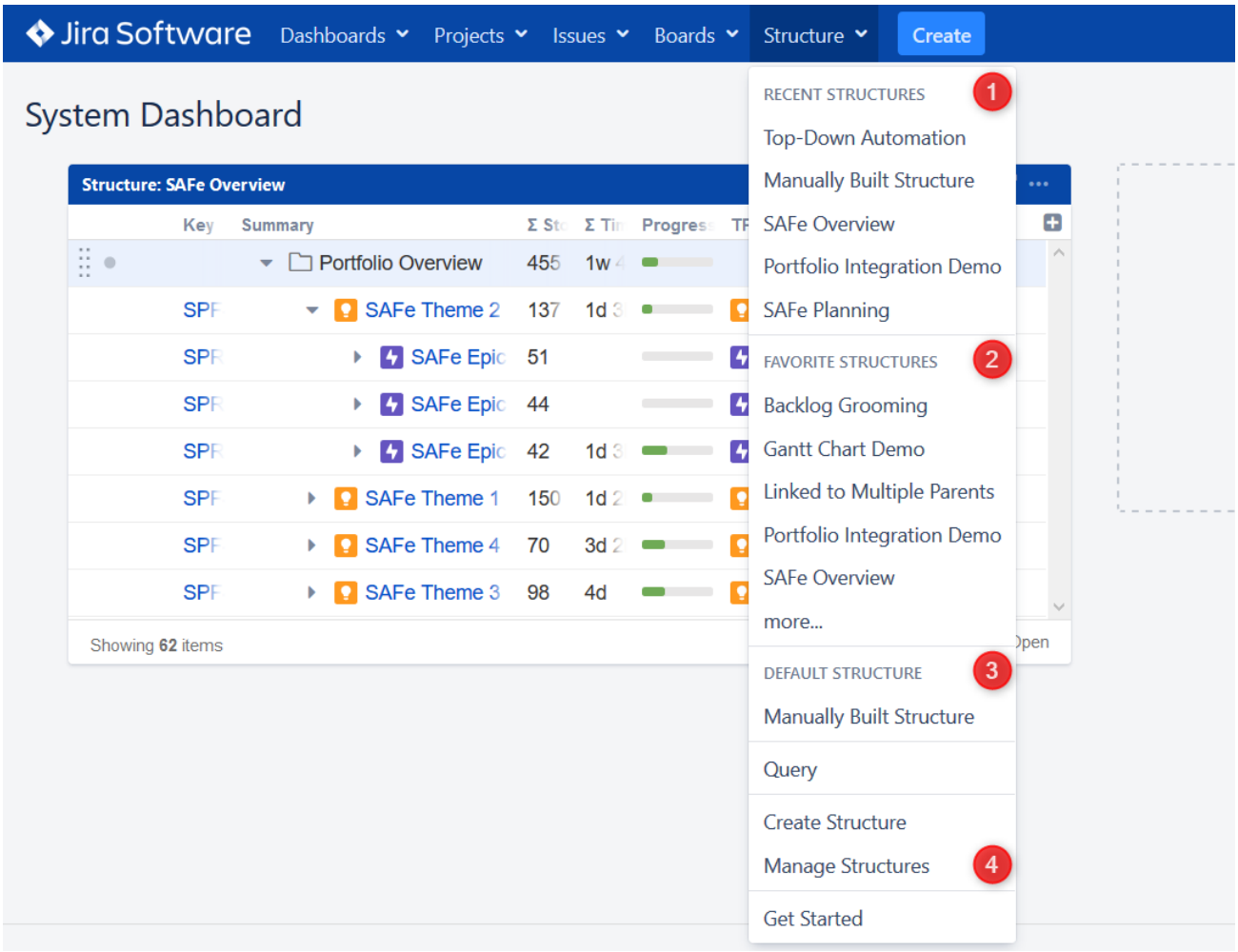
Once you install Structure, you will see a new item added to the top-level navigation bar (1).

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).



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:

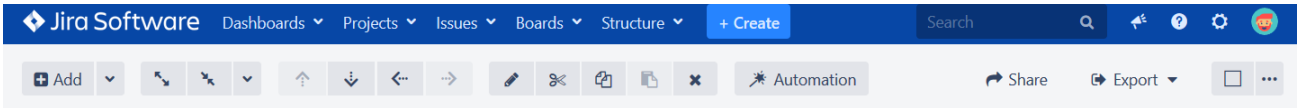


The menu has several sections:

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(see page 357). If a separate default structure is defined(see page 482) for the current project, it will be listed here as well.
4. **Manage Structures.** Takes you to the Manage Structures(see page 354) 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).

3.1.3 Main Structure Toolbar

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

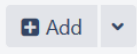
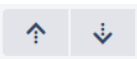


As you move the mouse pointer over the buttons in the toolbar, the active buttons are highlighted. Some buttons in the toolbar may be grayed out, indicating that these actions are not currently possible. For example, you cannot use the Paste action unless you have items in your clipboard, so the button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

3.1.3.1 Available Actions

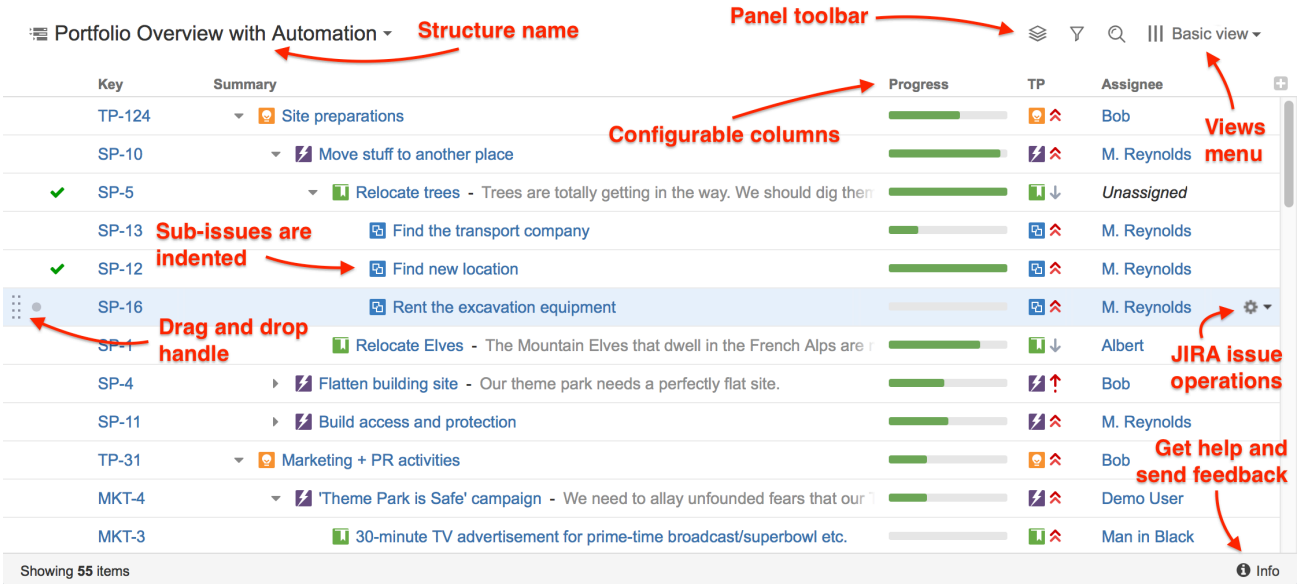
The following actions are accessible from the Structure toolbar.

Button	Action	More Information	Keyboard Shortcut
	Create a new item and add it under the item currently selected in the structure. By default, you can add either new issues or new folders (Confluence pages are available if you have Structure.Pages installed). To search for existing issues, click the drop-down menu next to the Add button.	Creating new items	**Enter**
	Expand or collapse the whole hierarchy. Use the drop-down menu to expand the hierarchy to a specific level.	Navigating Structure(see page 73)	**++** / **--**
	Move an item up or down within the structure, without changing the item's parent. If it is not possible to move an item up or down without changing its parent, the respective option will be grayed out.	Moving Issues Within Structure(see page 99)	**Ctrl+Up** / **Ctrl+Down** or **Command+Up**/ **Down**
	Unindent / Indent the item one level. If either or both of these moves are not possible, the option(s) will be grayed out.	Moving Issues Within Structure(see page 99)	**Ctrl+Left** / **Ctrl+Right** or **Command+Left**/ **Right**
	Edit the currently selected issue / stop editing and save changes.	Editing Issues(see page 114)	**Tab**

Button	Action	More Information	Keyboard Shortcut
	Cut the selected item(s) to the clipboard(see page 352).	Issue Clipboard(see page 352)	**Ctrl+x** or **Command+x**
	Copy the selected item(s) to the clipboard(see page 352).	Issue Clipboard(see page 352)	**Ctrl+c** or **Command+c**
	Paste item(s) from the clipboard(see page 352) into the structure.	Issue Clipboard (see page 352)	**Ctrl+v** or **Command+v**
	Remove the currently selected item(s) from the structure.	Removing Issues(see page 107)	**Delete**
	Switch Automation editing mode on/off.	Automation(see page 122)	**~**
	Create a perspective(see page 92) link to share the current view.	Perspective(see page 92)	
	Open a printable page(see page 380) with the structure or export the structure to Excel(see page 378).	Printing(see page 380) Exporting to Excel(see page 378)	
	Select the layout options, including opening the secondary panel(see page 350).	Secondary Panel(see page 350)	

3.1.4 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(see page 331) to the Structure panel (or by selecting one of the predefined views).



Within the Structure widget, you can:

- Rearrange issues and adjust their position within the hierarchy
- Edit issues
- Perform Jira issue operations
- Search and filter issues
- Switch to a different structure(see page 80)

To learn more about performing each of these actions, see Moving Items within Structure(see page 99) and Working with Issues(see page 108).

> Structure widget is displayed on the Structure Board(see page 83), Issue Page(see page 85) and in other places in JIRA(see page 83).

3.1.4.1 Navigating Between Items

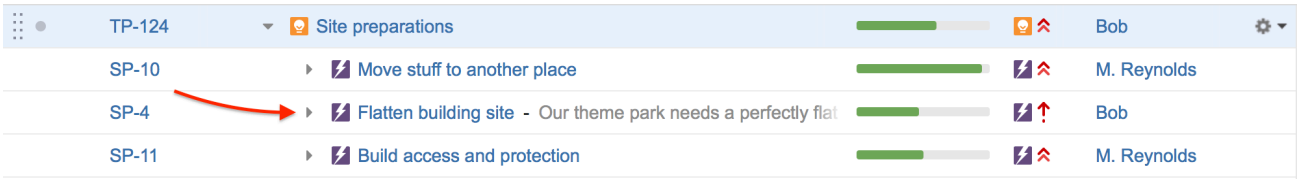
Navigating with Mouse

Using your mouse, you can:

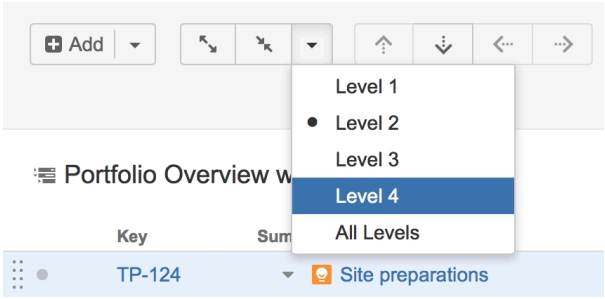
- Select an item by clicking anywhere in the item's row, except on a link
- Scroll up and down
- View the Issue Details(see page 108) panel by clicking the Key or Summary link (to customize this behavior, see Viewing Issue Details(see page 108))
- Show or hide sub-items

Show/Hide Sub-Items

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



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



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

Navigating with Keyboard

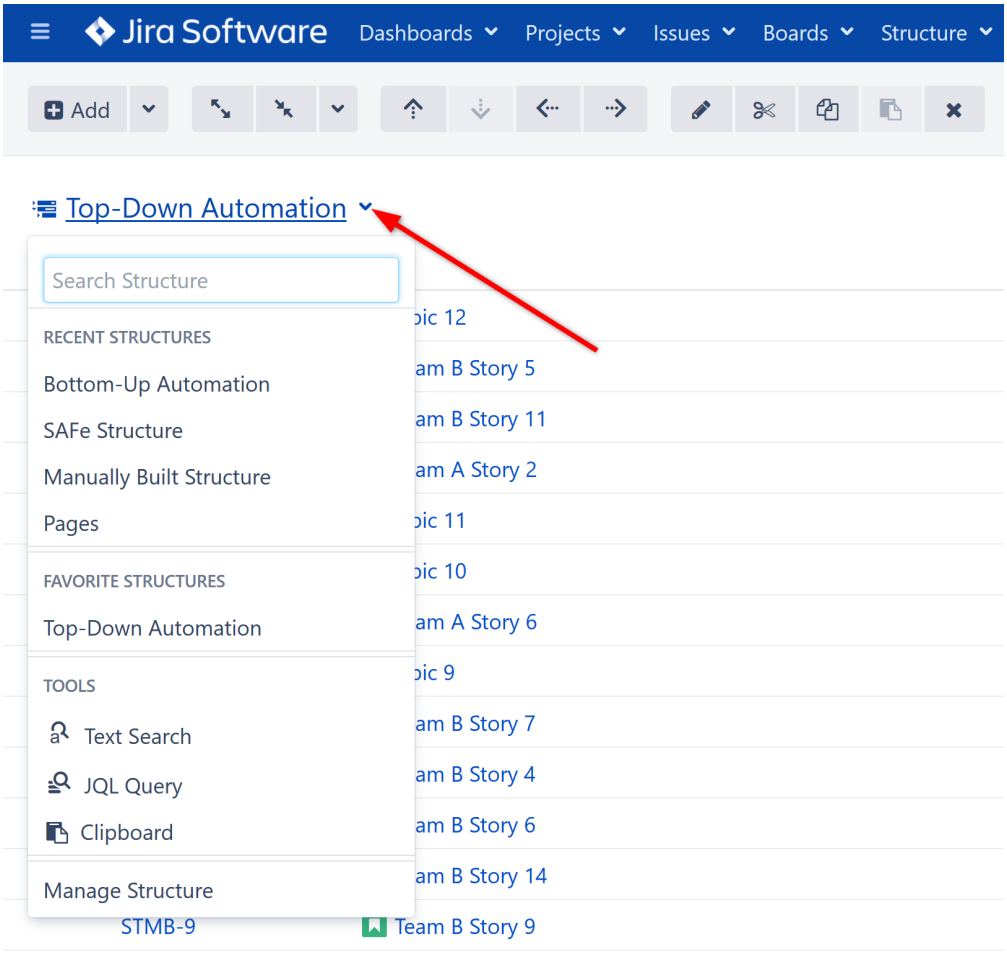
Using the keyboard, you can:

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

There are dozens of keyboard shortcuts(see page 384) available to simplify your Structure experience. Press **Ctrl+?** to view the shortcuts cheat sheet, or click **Info** at the bottom of the structure widget.

3.1.4.2 Switching Between Structures

If you need to switch structures without leaving the current Structure widget, instead of using the Structure menu(see page 74), click the name of the structure inside the widget:



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

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

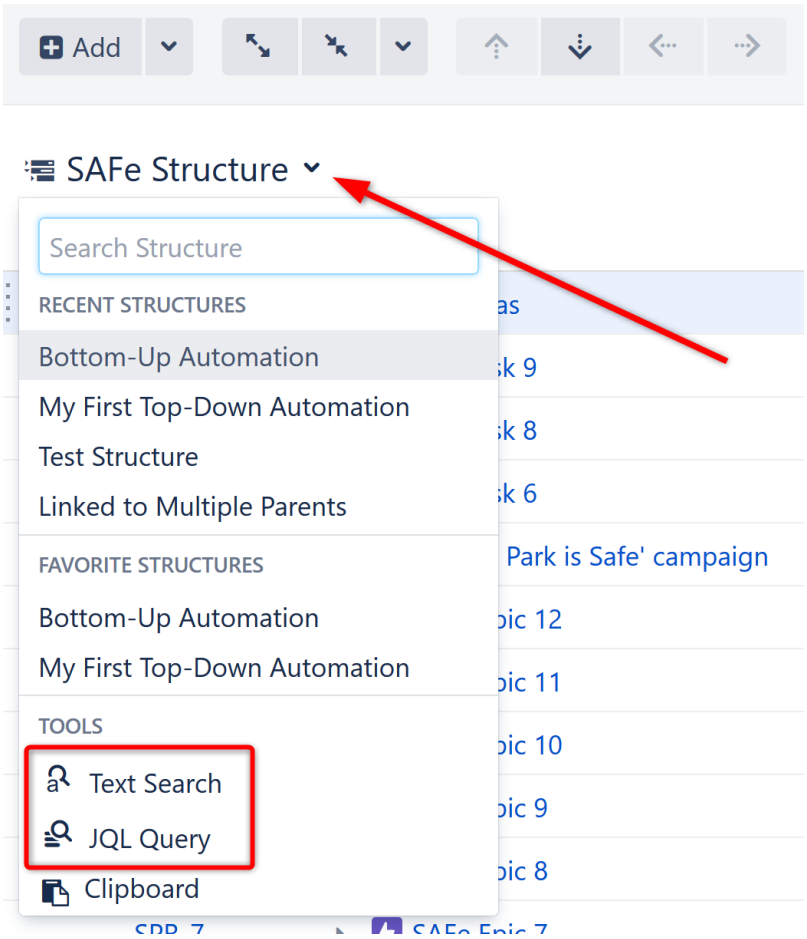
> Apart from choosing structures, you can also use this drop-down to run text and JQL searches, see the clipboard contents and - if you have Structure.Pages installed - search for Confluence pages. All of these options are available under the Tools section of the menu, when using Structure Board(see page 83) or Structure on the Project Page(see page 90)

3.1.4.3 Using Structure Widget for Searching

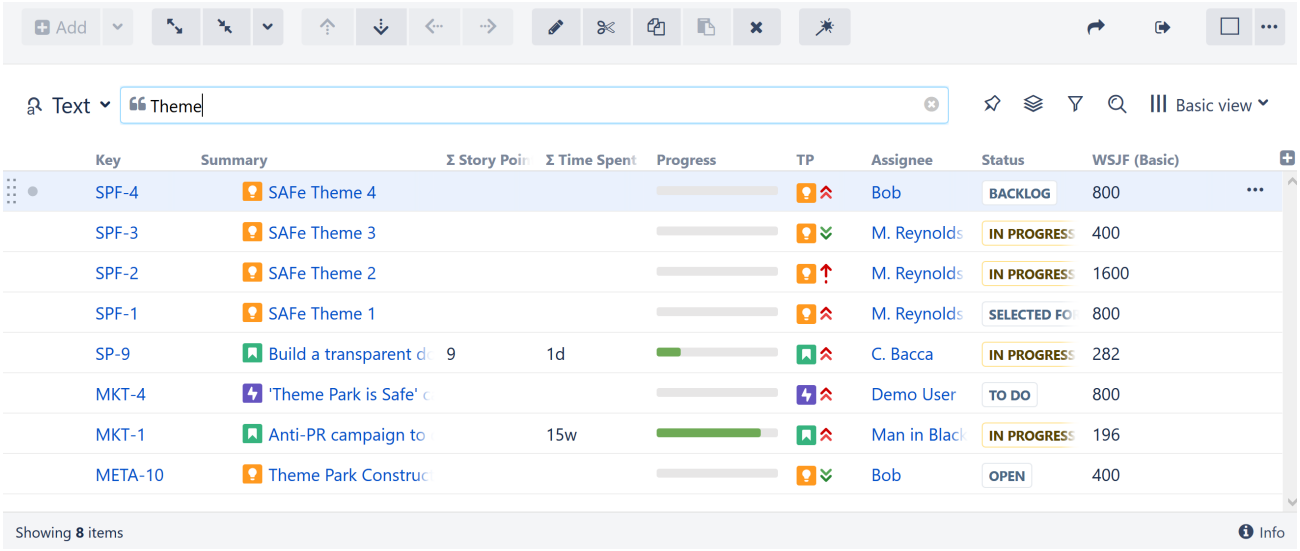
On the Structure Board you can use the structure widget not only for showing structures, but also for searching existing issues (using JQL or text search) and displaying your clipboard contents.

To search existing issues:

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



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



Just like when you're working with a structure, you can select a specific **view** for your search results and then add and arrange columns(see page 331) as necessary.

The **structure panel toolbar** also works for search results the same way it works for structures. You can apply sorting(see page 81), additional filtering(see page 183) and more complex transformations(see page 184).

⚠ Search only looks for issues from structure-enabled projects(see page 477).

3.1.5 Jira Pages with Structure

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

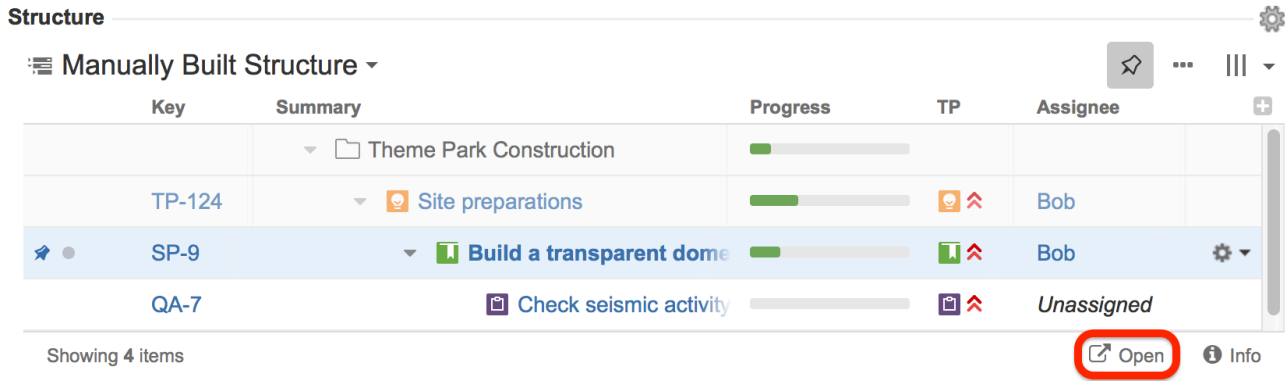
- On a dedicated Structure Board
- On the issue page
- On project pages and agile boards
- On a dashboard

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(see page 83)
- Structure on the Issue Page(see page 85)
- Structure on the Project Page(see page 90)
- Structure on Agile Boards(see page 91)
- Structure Gadget(see page 394)

3.1.5.1 Open on Structure Board

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



The structure will open on Structure Board with the same filters(see page 183) and transformations(see page 184) that were applied in the original gadget. For examples, if your Dashboard Gadget is configured to only show items from a specific project, sorted by Assignee, that's exactly what you'll see on the Structure Board.

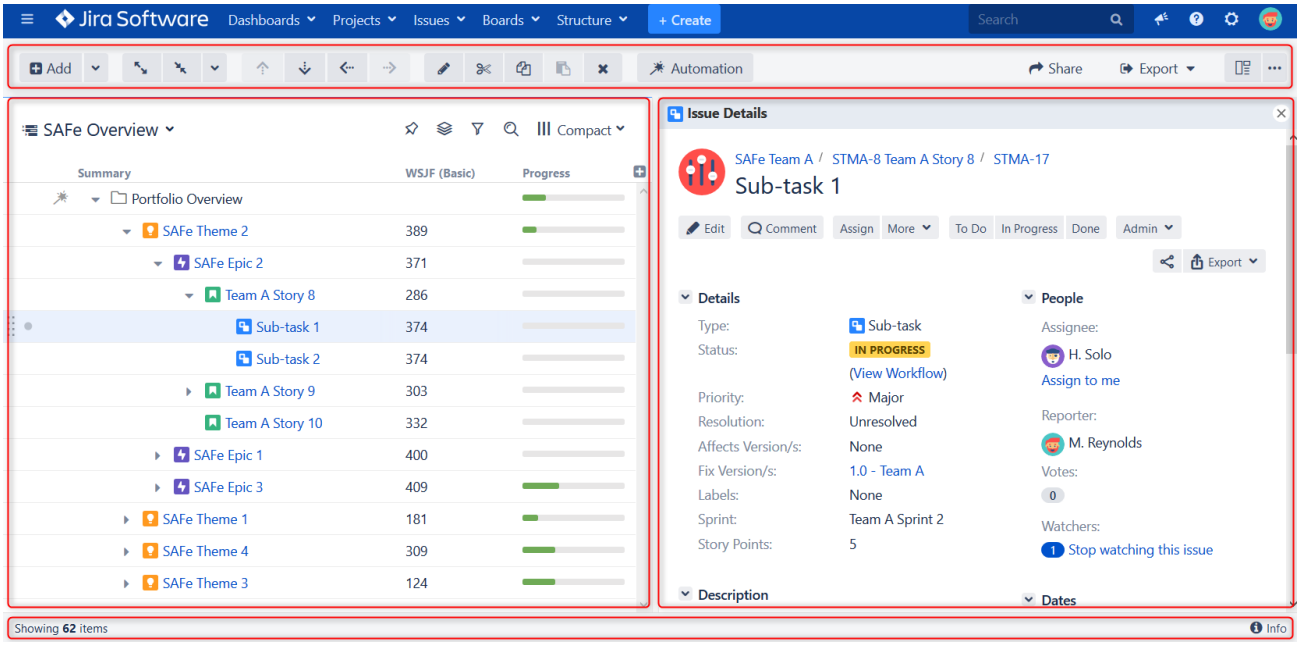
To review or remove these transformations, click the Transformations button in the panel toolbar.

3.1.5.2 Structure Board

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

The main elements of the Structure Board are:

- **Structure Toolbar**. At the top of the Structure Board, the Toolbar(see page 76) gives you access to the main functions for building and working with structures.
- **Working Panels.** The left panel always displays the structure widget (see page 78)or search results(see page 81), while the left panel can display another structure widget, issue details(see page 108), history(see page 375) and other features based on the add-ons you have installed.
- **Status Bar.** At the bottom of the Structure Board, this shows the number of items currently displayed, links for the **Undo** operations and notifications.



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

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

- At the bottom of the Structure menu, select Manage Structure(see page 354). From the Manage Structure page, you can browse and search for available structures.
- Open another structure and switch between structures (see page 80)on the Structure Board.
- If you know the structure ID, you can open it directly with a URL:
  `http://your.jira.address/secure/StructureBoard.jspa?s=structure-id`

Keyboard Shortcut

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

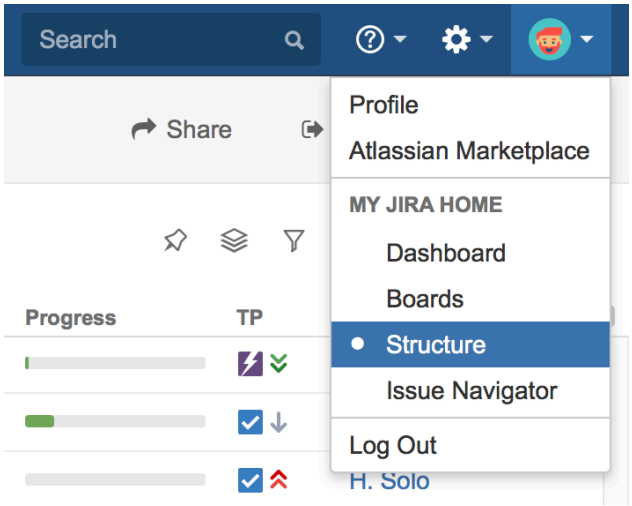
Save Time!

You can make the Structure Board your Jira Home page(see page 85).

Making Structure Board Your Jira Home

If you want to go straight to the Structure Board(see page 83) when you log in to Jira, you can make it your Jira Home page. To do so:

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

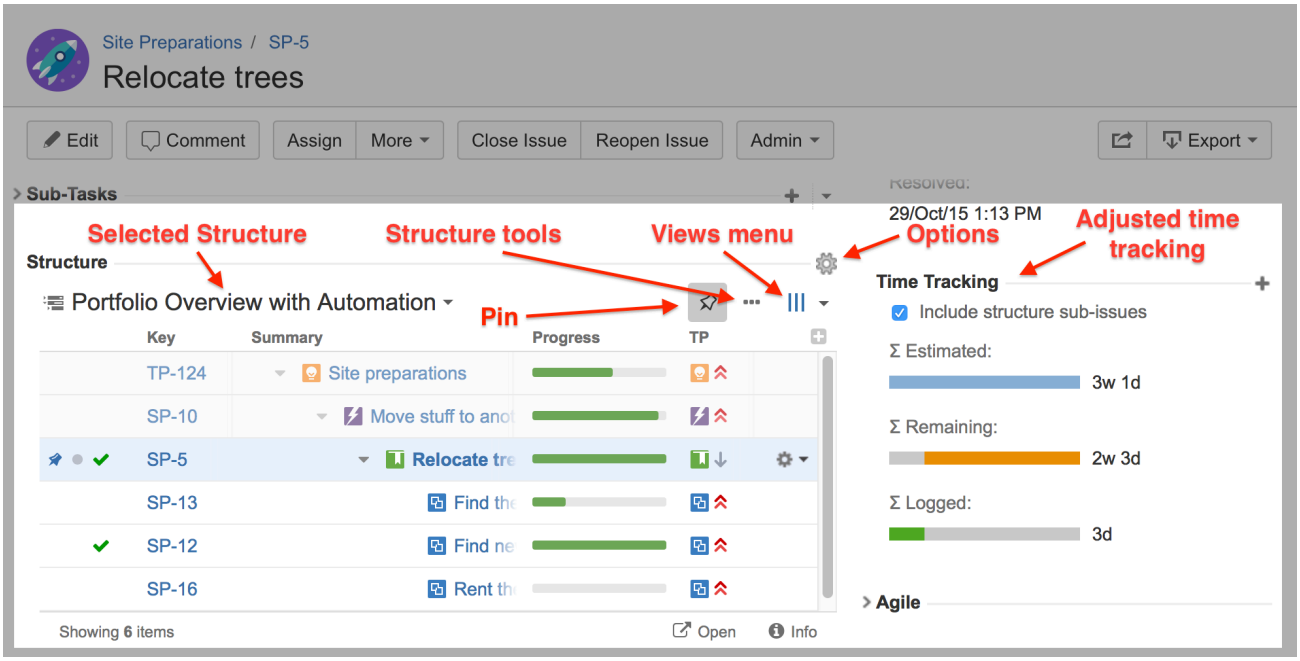


When used as a Jira Home page, the Structure Board will show your most recently opened structure.

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

3.1.5.3 Structure on the Issue Page

If an issue belongs to a project for which the Structure add-on is enabled(see page 477), the Structure widget is displayed on the issue details page. The widget is presented as a separate section, located right above the **Activity** section.



When you open an issue page, the structure that appears there is based on the Structure Options for the Issue Page(see page 88).

Pinned Issue

The issue itself is automatically located and Pinned(see page 192) in the structure. This means only the parent issues and sub-issues of the viewed issue are displayed.

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

Structure widget can be hidden from the Issue Details page. Please refer to the Structure Administration(see page 498) article for details.

Starting with Jira 6, search results on the Issue Navigator page can display the details of a selected issue in a side panel. This details panel also contains a Structure section. Since the details panel is often much narrower than the issue page, it may be helpful to configure a view(see page 336) to fit only the necessary information in the smaller space left for the Structure widget.

Unique Features

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

- Collapsing/Showing Structure Section(see page 87)
- Structure Selection(see page 87)
- Adding Issue to a Structure(see page 87)
- Structure Tools(see page 88)
- Views and Options Drop-Downs(see page 88)
- Adjusted Time Tracking Section(see page 88)

- Activity Tab(see page 88)

Collapsing/Showing Structure Section

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

Also, the Structure section is automatically hidden if the issue you open does not belong to the selected structure. (This behavior can be adjusted in the Structure Options(see page 88).) When the Structure section is hidden, the issue hierarchy is not loaded from the server – it will be loaded only when you first open the Structure section.

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

Structure Selection

As you open the issue details for the first time, you will see one of the structures that contain this issue (this behavior can be adjusted through Options(see page 88)).

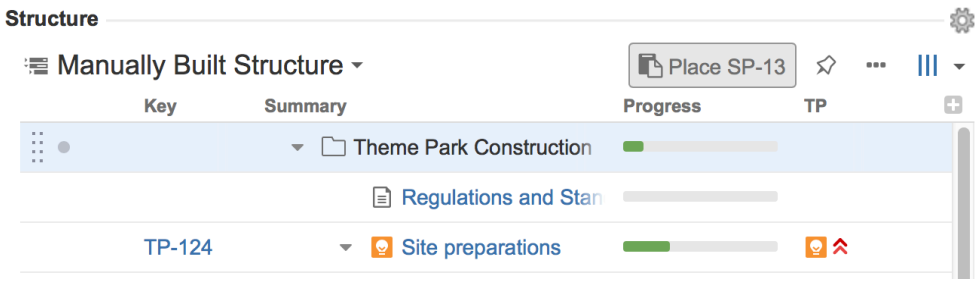
To switch to a different structure, simply click the name of the currently displayed structure and select the one you want to see. You will see those structures that contain this issue in the top section of the displayed menu.

> Structures where this issue is added through Automation(see page 122) will not appear in the list of structures containing the issue, as this would significantly affect performance.

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

Adding Issue to a Structure

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



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

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

Structure Tools

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

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

Views and Options Drop-Downs

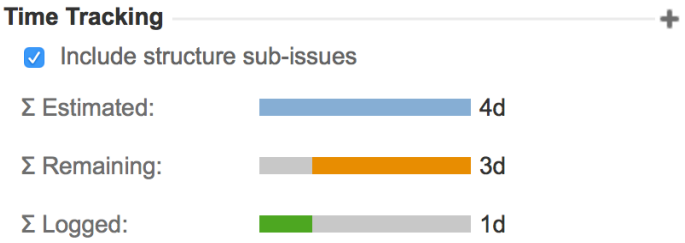
Located at the right corner of the Structure section header are Views and Options icons.

- Click Views icon to open the Views Menu(see page 337) and select another view for the displayed structure.
- Click Options icon to open Structure Options for the Issue Page(see page 88).

> An asterisk appears next to the view name if it has been locally adjusted(see page 339).

Adjusted Time Tracking Section

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



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

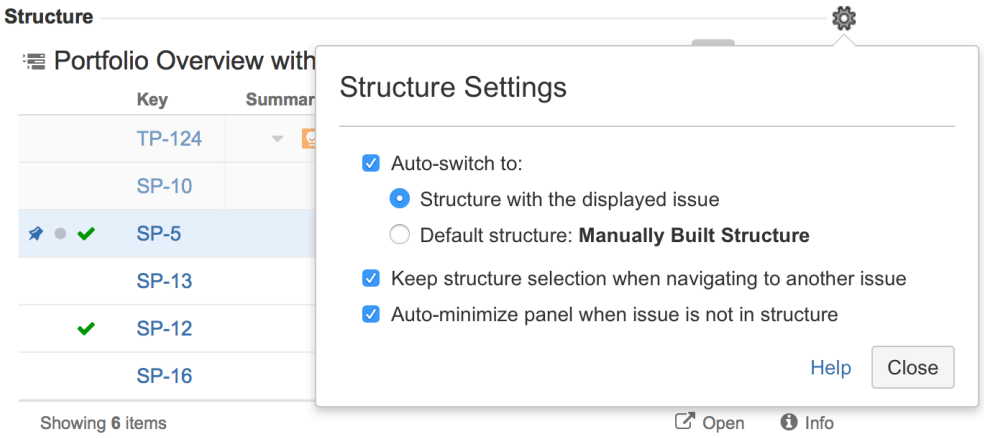
If the time tracking section is not present, it means that neither the current issue nor its sub-issues have any time tracking info.

Activity Tab

As you work with structures, all changes are added to the Jira Activity Stream. As such, all changes to the structure that affect the current issue will be displayed in the **Activity** tab of the issue page. This may be useful if you want to find out why this issue is in a particular position within a structure, including who added or moved it, and when. See Structure Activity Stream(see page 381) for more information.

Structure Options for the Issue Page

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



Which Structure is Displayed?

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

This is controlled by a number of parameters:

Auto-switch	When auto-switch is turned on, the structure is selected based on which project and structures the issue belongs to. When auto-switch is turned off, the Structure section shows the structure that the user opened last on the Structure Board (the *current* structure).
Auto-switch: structure with displayed issue	When this auto-switch mode is selected, Structure looks for a structure that contains the issue displayed on the page.
Auto-switch: default structure	When this auto-switch mode is selected, the Default Structure(see page 357) for the issue's project will always be selected (even if the issue is not in that structure).
Keep structure when navigating	When you click on another issue within the Structure widget, the browser takes you to that issue's page. If this option is turned on, the new page displays the same structure as the page you navigated from (auto-switch is not applied). We recommend leaving this option on. This will prevent you from unintentionally switching structures while reviewing a structure's issues.

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

Auto-Minimize?

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

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

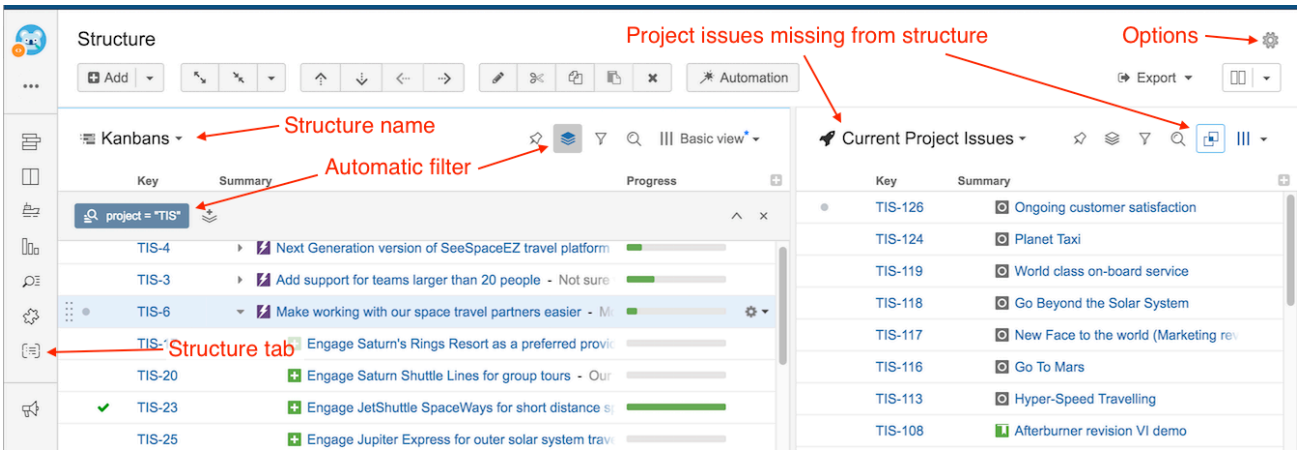
Options Scope and Default Options

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

✓ The default values of these options can be configured by the Jira Administrator on the Structure Defaults(see page 481) page.

3.1.5.4 Structure on the Project Page

If a project is enabled for Structure(see page 477), you will see a new Structure icon on the side navigation bar. Clicking this will open the Structure widget on the Project Page.



This is a fully-functional Structure widget and has the same functionality you can find on the Structure Board(see page 83), 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.

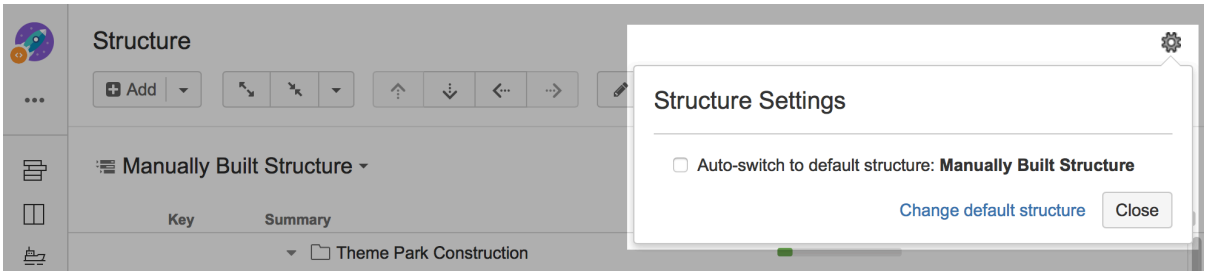
The primary (left) panel displays the most recently viewed structure or the default structure for the current project (as defined in the Options dialog). You can quickly switch to another structure by clicking the structure name and selecting the desired structure.

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.

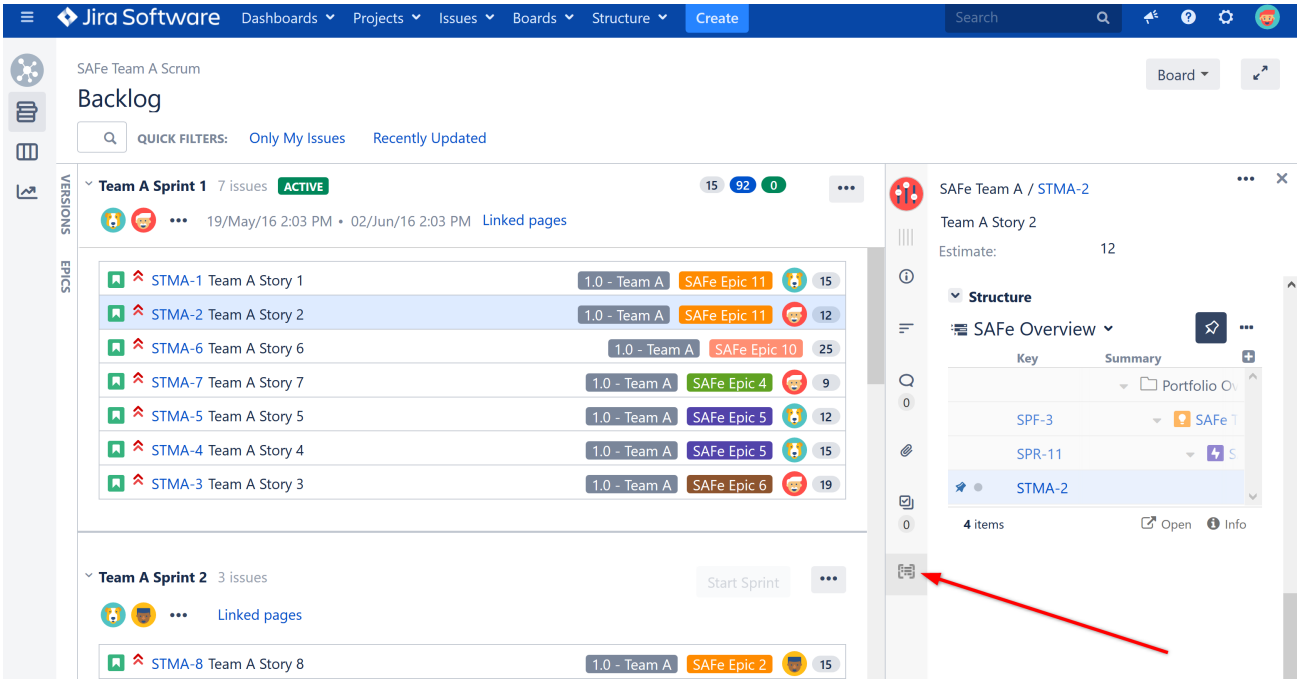
The default value for this option can be configured by the JIRA Administrator on the Structure Defaults(see page 481) page.

Perspectives are Unavailable

It is not possible to share a perspective(see page 92) from the project page.

3.1.5.5 Structure on Agile Boards

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



The Structure tab displays the standard Structure widget in Pinned Item Mode(see page 192) (highlighting the position of the selected issue and its sub-issues). You can unpin the structure by clicking the Pin button in the toolbar or hitting **Ctrl+.** (period).

Adding Columns

Due to the constrained horizontal space, Structure initially displays only the Key and Summary columns in the Agile tab. You can add more columns(see page 331) by clicking the Add Columns button (the plus sign).

Switching Issues

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

Editing Issue Fields

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

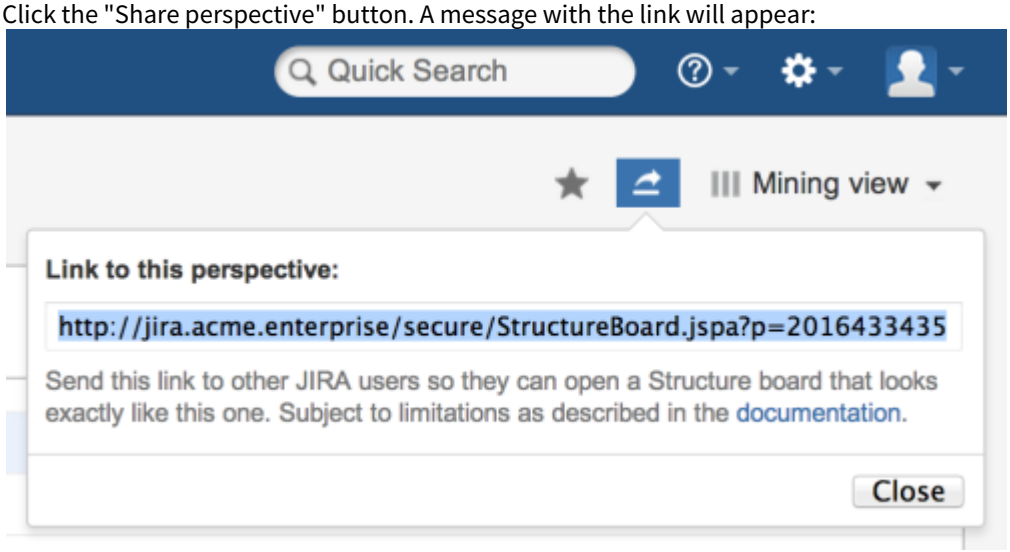
> Only the users who have access to Structure(see page 478) will see the Structure tab on Agile boards.

3.1.6 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(see page 83)
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(see page 341) 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(see page 490).
5. Once created, a perspective becomes accessible to any person who has access to the structure for which that perspective was created.

3.2 Basic Operations

- Adding Items to a Structure(see page 94)
  - Adding Issues(see page 94)

        - Folders(see page 95)
        - Memo(see page 97)
- Moving Items within Structure(see page 99)
    - Selecting Multiple Items(see page 101)
    - Using Cut, Copy and Paste(see page 103)
        - Moving Items Between Structures(see page 103)
        - Moving Items Within a Structure(see page 104)
    - Using Drag and Drop(see page 105)
- Removing Items from Structure(see page 107)
- Undoing Changes(see page 107)
- Working with Issues(see page 108)
    - Viewing Issue Details(see page 108)
    - Creating New Issues(see page 110)
    - Editing Issues(see page 114)
        - Changing Fields(see page 115)
        - Keyboard Shortcuts in Edit Mode(see page 117)
        - Correcting Input Errors(see page 119)
        - Editing from Gadget(see page 119)
        - On E-mail Notifications(see page 120)
    - Bulk Change(see page 120)
    - Cloning Multiple Issues(see page 121)
    - Using Jira Actions(see page 121)

3.2.1 Adding Items to a Structure

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

- Issues(see page 94)
- Folders(see page 95)
- Memo(see page 97)
- Generators (Automation)(see page 122)
- Effectors(see page 154)

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

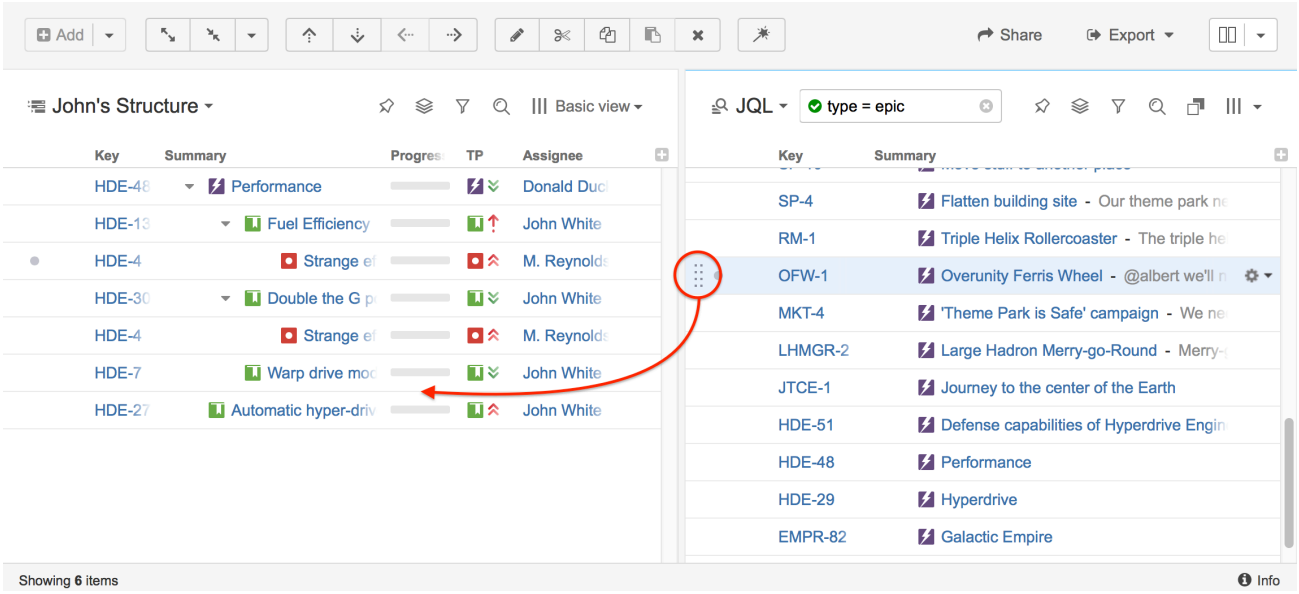
3.2.1.1 Adding Issues

Issues can be added to a structure from the Structure Board(see page 83) or their own Issue Page(see page 87).

Adding issues from Structure Board

On the Structure Board, open the secondary panel(see page 350) and search for existing issues using the text or JQL search. You can import issues one at a time, or select multiple items(see page 101).

After selecting the issue(s) you need, add them to your structure using drag-and-drop(see page 105) or copy/paste(see page 103).

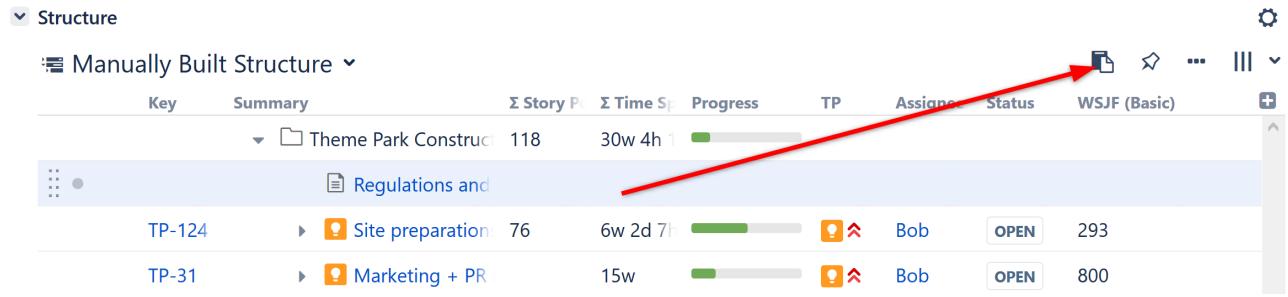


You can also add issues from another structure. Simply open the structure in the secondary panel(see page 351), and use drag-and-drop or copy/paste to import issues.

Adding issues from the Issue Page

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

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



Issues can be added and removed automatically using Automation(see page 122).

3.2.1.2 Folders

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

Fun with Folders — Basic view

Key	Summary	Σ Story Poin	Σ Time Spe	Progress	TP	Assignee	Status	WSJF (Basic)
	Project A	46	2d					
STMA-1	Team A Story 1	15	1d			Anna M.	IN PROGRESS	317
STMA-36	Task					*Unassigned*	TO DO	800
STMA-2	Team A Story 2	12	6h			Anna M.	IN PROGRESS	309
STMA-3	Team A Story 3	19	2h			Anna M.	IN PROGRESS	286
	Project B	23	2h					
STMB-1	Team B Story 1	19	2h			Mary (Inactive)	TO DO	306
STMB-43	Bug 1		2h			*Unassigned*	DONE	
STMA-35	Bug 2					*Unassigned*	TO DO	800
STMB-2	Team B Story 2	4				Mary (Inactive)	IN PROGRESS	342

Some common uses for folders include:

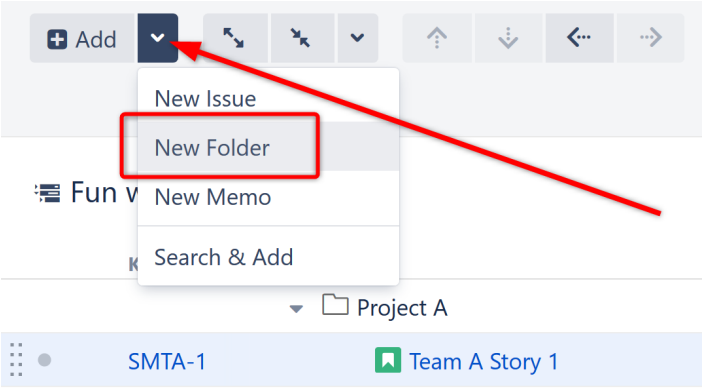
- Organizing issues into specific categories
- Separating different projects or different parts of a project
- Creating different Automation(see page 122) rules for different parts of a structure (if a generator is placed beneath a folder, it will only affect items in that folder - see Generator Scope(see page 169))
- Placing a structure within a structure (it's not necessary to use folders, but we recommend it)

Group generators(see page 141) make their own folders to group items by a common attribute.

Adding Folders to a Structure

To add a folder to a structure:

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

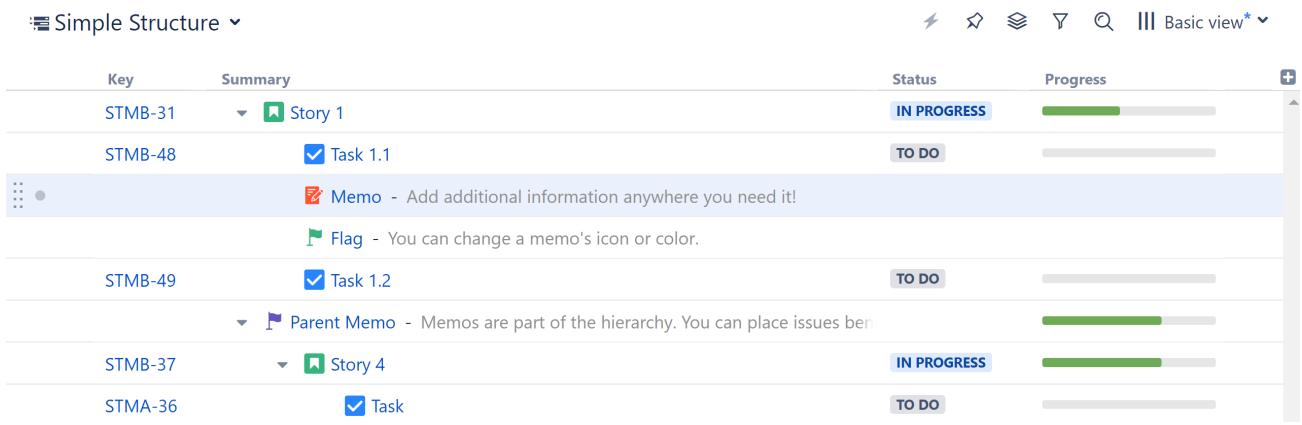


Keyboard shortcut: Press **Enter** to open the Add New Item dialogue, and press **Alt+Up/Down** to select between Issue, Folder and Memo.

Users must have Edit permissions(see page 362) or higher to add folders to a structure.

3.2.1.3 Memo

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



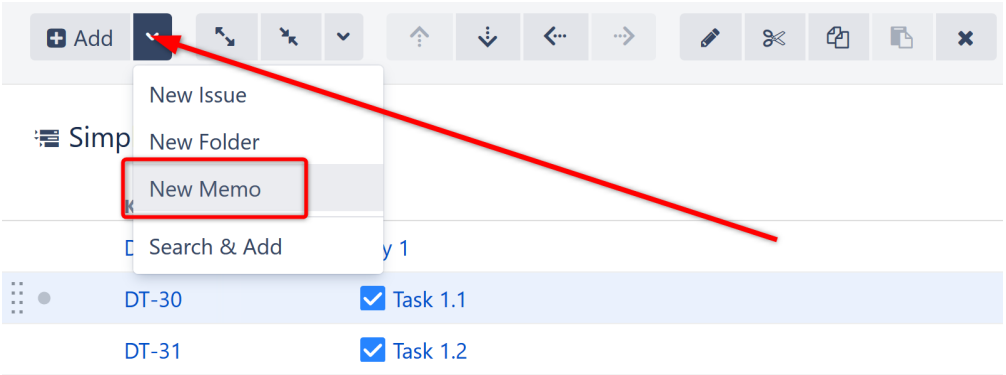
Memos can serve a variety of purposes within a structure:

- Add notes or reminders that pertain to the structure or project as a whole, rather than just a single issue (for that, try a Notes column(see page 319))
- Add high-level requirements directly to your structure
- Add a placeholder for other items
- Use them like folders, grouping issues within your hierarchy (with the added benefit of color and text)
- Just about anything else you can think of!

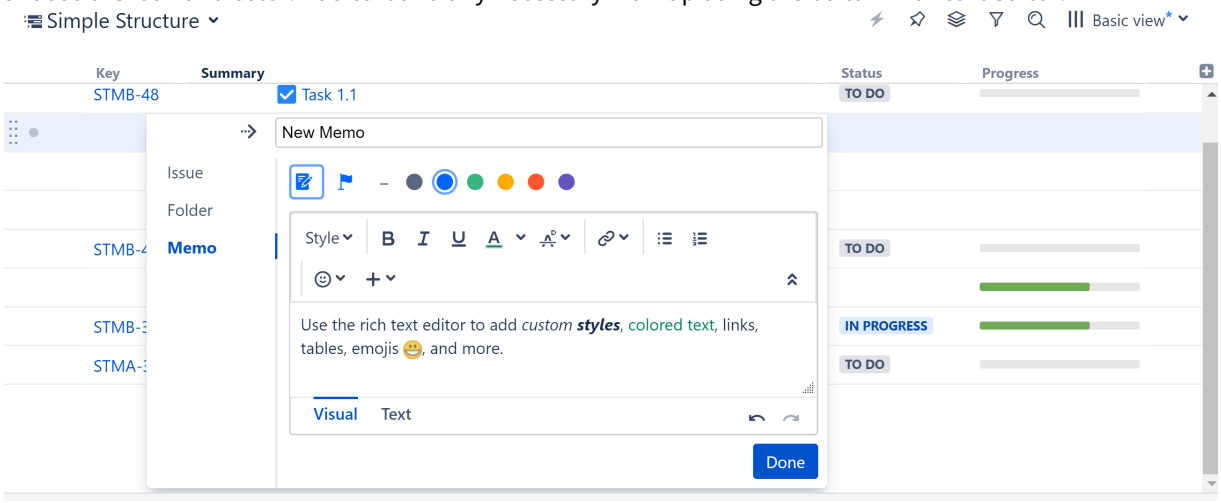
Adding a Memo to a Structure

To add a memo to a structure:

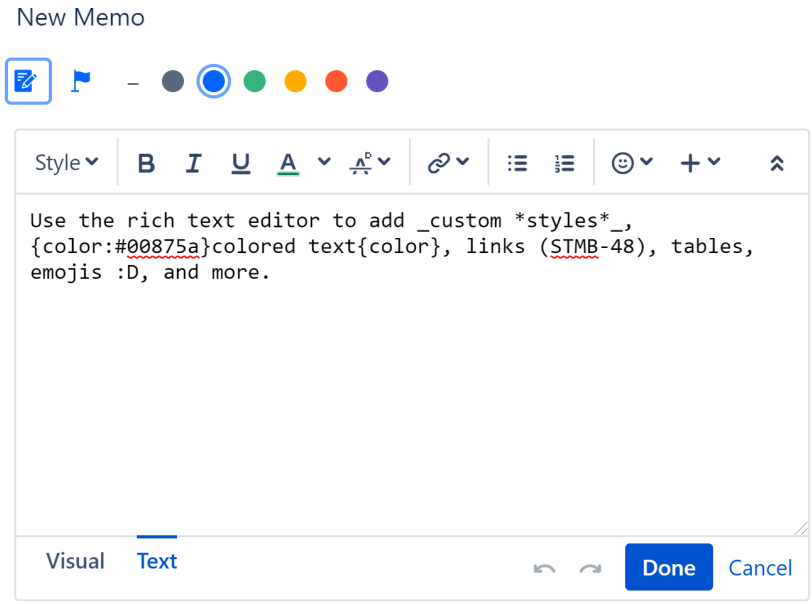
1. Select the memo's location (the memo will be placed at the same level in the hierarchy, beneath the currently-selected item).
2. Open the **Add** drop-down menu.
3. Select **New Memo.**



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



The editor opens in Visual mode, a WYSIWYG editor. You can also edit text using wiki markup by switching to the Text mode - see Jira's Text Formatting Notation Help page[12] for a complete list of available formatting options and conventions.

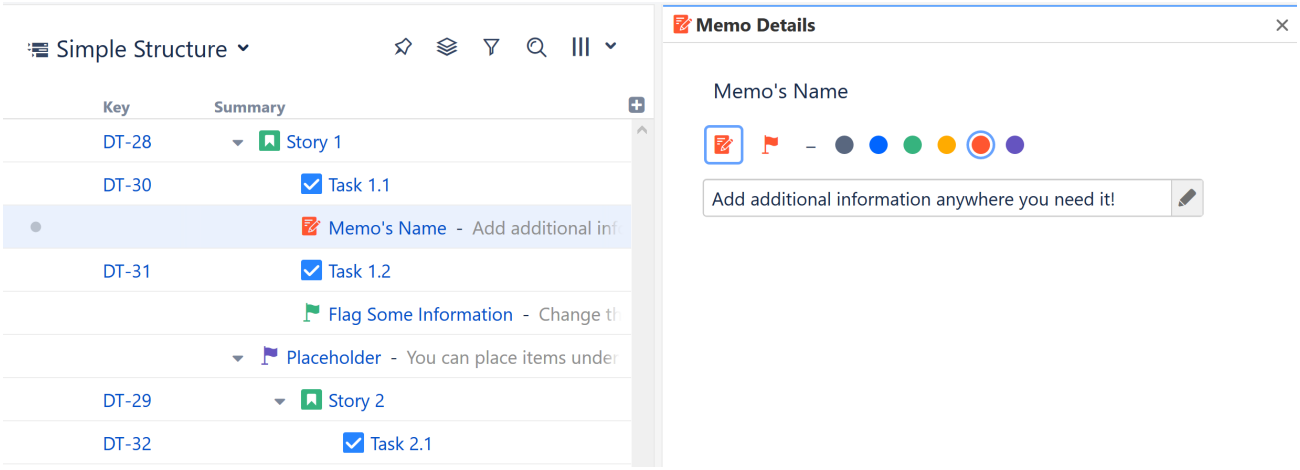


Keyboard shortcut: Press **Enter** to open the Add New Item dialogue, and press **Alt+Up/Down** to select between Issue, Folder and Memo.

Editing a Memo

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

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



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

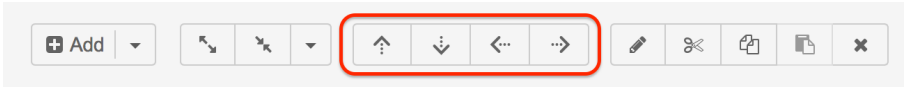
The memo name can also be edited directly from the structure by clicking the edit icon in the toolbar or using the **Tab** or **F2** keyboard shortcut.

3.2.2 Moving Items within Structure

3.2.2.1 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.

Operation	Keyboard Shortcut	What it does
Level Down / Indent	Ctrl + Right	Move the item to be a sub-issue of its current preceding sibling.

Mac Users: Use Cmd instead of Ctrl.

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

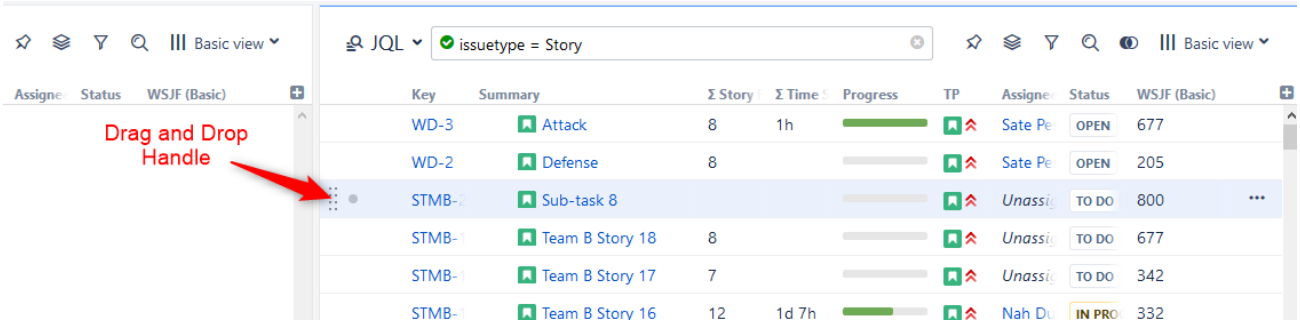
3.2.2.2 Advanced Moves

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(see page 105) 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(see page 103) 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(see page 107).

3.2.2.3 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:

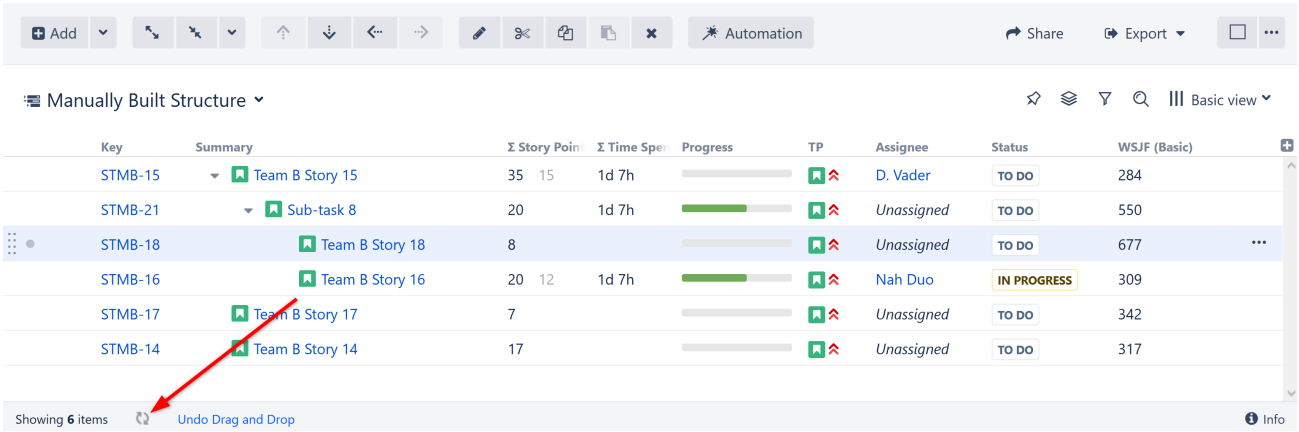
- Moving items with the toolbar or keyboard (Move Up/Down and Level Up/Down) only works if all of the selected items are at the same level in the hierarchy and under the same parent.
- Drag-and-Drop and Copy & Paste support multiple-item moves in any configuration.

See also: Selecting Multiple Items(see page 101)

3.2.2.4 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.



3.2.2.5 Moving Items with Automation

If you used Automation(see page 122) 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.

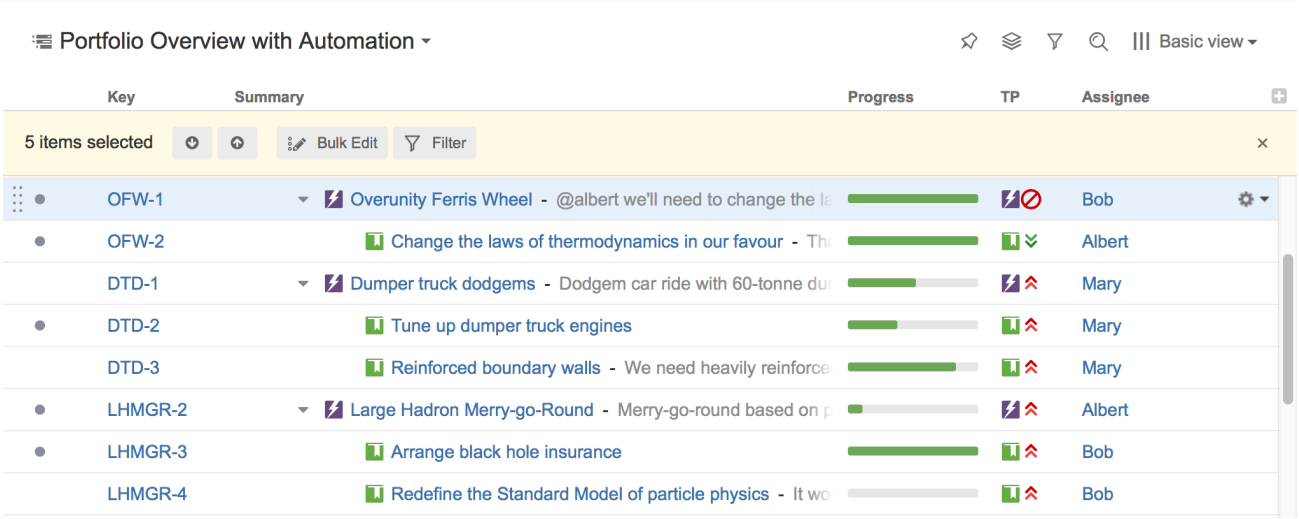
In order to move items freely within a generated structure, you may need to enable Manual Adjustments(see page 175).

3.2.2.6 Selecting Multiple Items

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

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

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



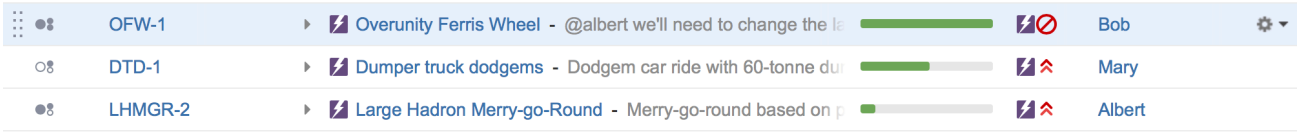
The selection panel offers the following features:

- Move focus from one selected item to another by clicking the up and down arrows
- Bulk Edit(see page 120) the selected issues using the Jira bulk change wizard
- Show only selected items and their parents by clicking the Filter button
- Remove all selections by clicking the close button in the right corner of the panel

Special Selection Markers

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

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



The meaning of each marker is as follows:

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

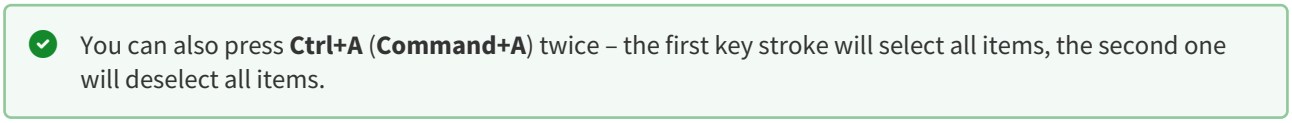
Changing Multiple Items

The following actions work with the multi-selection:

- Drag and drop(see page 105) lets you move a selection of items within a structure or between two structures
- Cut and paste(see page 103) allows you to move items within a structure and between different structures
- Remove button(see page 107) or **Delete** key lets you remove multiple items from the structure
- Toolbar buttons *Move Up*, *Move Down*, *Indent* and *Outdent* are allowed for multiple items, only if all items in the selection are at the same level in hierarchy and have the same parent item
- Bulk Change(see page 120) lets you use the Jira bulk change wizard to modify the selected issues

Exiting Multi-Select Mode

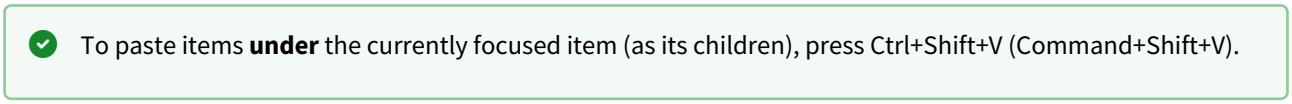
To exit multi-select mode (and deselect all items), press the **x** button at the far right of the selection panel or press the **Escape** key.


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

3.2.2.7 Using Cut, Copy and Paste

While drag and drop(see page 105) is a nice and visual way to rearrange items, it is not always effective for dragging drag items across large structures. Copy and paste solves this problem: once you copy items to the clipboard, you can scroll through the structure until you find exactly where you want to paste them.

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

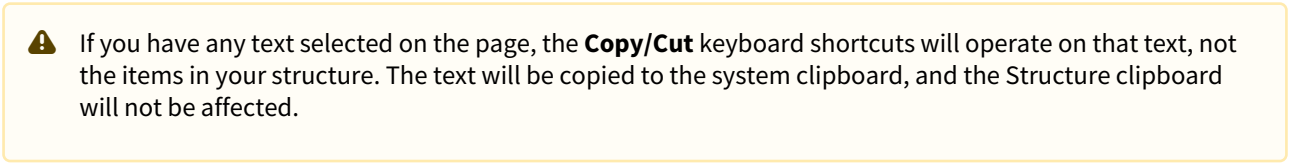

To paste items **under** the currently focused item (as its children), press Ctrl+Shift+V (Command+Shift+V).

See Selecting Multiple Items(see page 101) for details about multi-selection.

Copy / Paste Scenarios

There are two main scenarios for using the Issue Clipboard(see page 352):

- Moving Items Between Structures(see page 103)
- Moving Items Within a Structure(see page 104)


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

Moving Items Between Structures

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

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

Step 1 - Cut/Copy

First add the desired items to the clipboard:

1. Open the structure you wish to cut/copy from.
2. Select the items you want to cut or copy. Either select a single item, or use multiple select(see page 101).
3. Click the **Cut/Copy** button on the structure toolbar (or press **Ctrl+x/Ctrl+c** or **Command+x/Command+c**).
4. The selected items will be added to the clipboard and marked with a small scissors icon for cut or a clipboard icon for copied .

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

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

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

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

Step 2 - Paste

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

1. **In the same browser window**, switch to a desired structure (you can use Structure Board or any other JIRA page with Structure(see page 83)).
2. In the structure grid, select the item after which the items from the clipboard should be placed.
3. Click the **Paste** button on the toolbar (or press **Ctrl+v** or **Command+v**) to place the items at the same indentation level. To place the items **under** the selected item (as its children), press **Ctrl+Shift+v** (**Command+Shift+v** on Mac).

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

The Paste operation can be undone(see page 107).

If you need to copy the same set of issues to several different structures, you can open the Issue Clipboard(see page 352) in the secondary panel(see page 350) and use drag-and-drop(see page 105) to move the issues, instead of Paste. In this case, the issues will not be removed from the clipboard.

Moving Items Within a Structure

You can use Copy/Cut and Paste to move items within a structure. This is particularly useful when moving several items across a large structure, where drag-and-drop(see page 105) can be more difficult to manage.

To move an item within a structure:

Copy/Cut

First add the desired items to the clipboard:

1. Select the items you want to move. Either select a single item, or use multiple selection(see page 101).
2. To move the item, click the **Cut** button on the structure toolbar (or press **Ctrl+x** or **Command+x**). To create a second copy of the item within the structure, use the **Copy** button (or **Ctrl+c** or **Command+c**).
3. The selected items will be added into the clipboard and marked with a small scissors icon (for cuts) or a clipboard icon (for copies).

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

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

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

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

Paste

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

1. To view the contents of your **Clipboard**, you can open it in the secondary panel(see page 350). However, this is not necessary.
2. In the structure, select the item after which the items from the clipboard should be placed.
3. Click the **Paste** button on the toolbar (or press **Ctrl+v** or **Command+v**) to place the items at the same indentation level. To place the items **under** the selected item (as the children), press **Ctrl+Shift+v** (or **Command+Shift+v** on Mac).

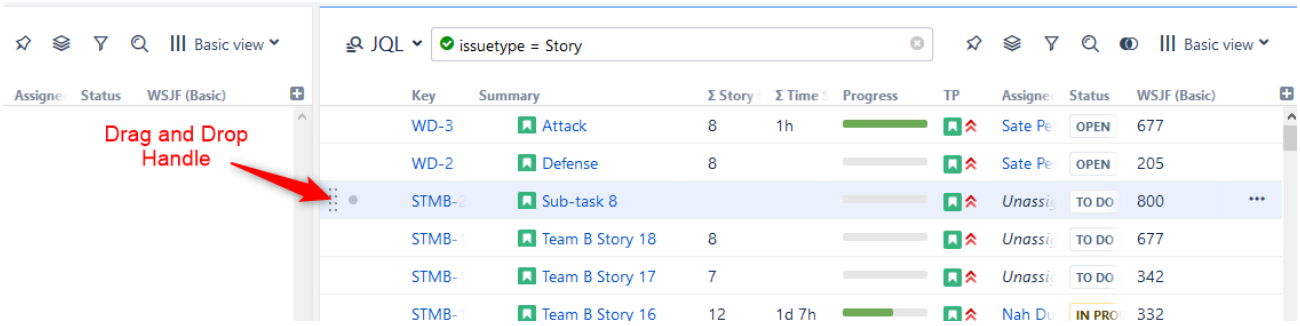
The Paste operation can be undone(see page 107).

3.2.2.8 Using Drag and Drop

Drag and drop allows you to quickly move or copy items within the structure or add items to a structure using the secondary panel(see page 350).

Moving Items

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



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

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

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

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

Copying Items

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

To copy an item from one structure to another (using Two-Panel Mode(see page 350)), simply drag and drop the item. If you want to move the item from one structure to another (so it's removed from the original structure) hold the **Ctrl** key as you drag it (**Alt** key for Mac).

Dragging Multiple Items

To move more than one issue at a time, first select multiple issues(see page 101) (click the gray dot at the beginning of each row) and then move them using the Drag and Drop Handle of any of the selected items.

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

Cancelling Drag

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

> Drag and drop can also be undone(see page 107).

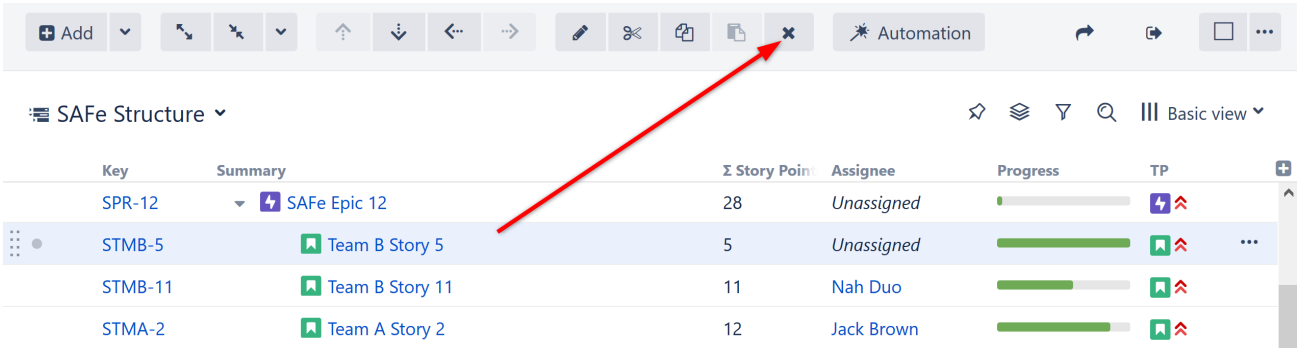
Scrolling Structure While Dragging

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

Using Cut & Paste(see page 103) may be more effective than Drag and Drop if you need to move items to a distant position.

3.2.3 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.



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

To delete more than one item, select the items(see page 101) and click **Delete**.

Removing items can be undone(see page 107).

3.2.4 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(see page 107) items from search results;
- Removing(see page 107) items from a structure;
- Drag-and-Drop(see page 105);
- The Paste operation of a Cut & Paste(see page 103) 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(see page 107), 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(see page 375).

3.2.5 Working with Issues

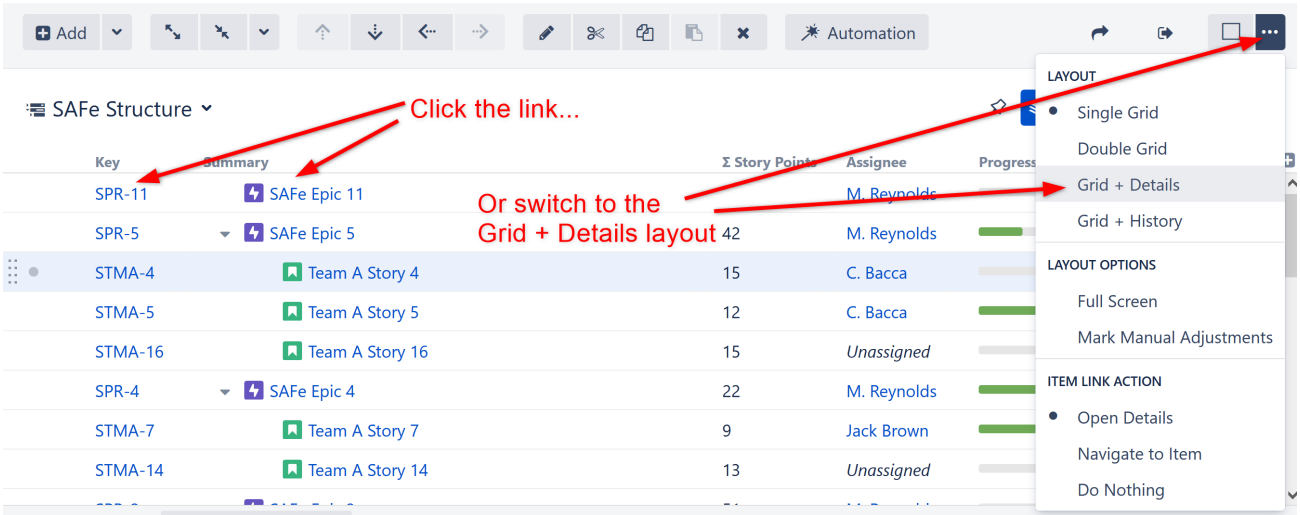
Structure allows you to view and edit issues without leaving the Structure widget.Select one of the following articles to see how you can work with issues from within a structure.

- Viewing Issue Details(see page 108)
- Creating New Issues(see page 110)
- Editing Issues(see page 114)
  - Changing Fields(see page 115)
  - Keyboard Shortcuts in Edit Mode(see page 117)
  - Correcting Input Errors(see page 119)
  - Editing from Gadget(see page 119)
  - On E-mail Notifications(see page 120)
- Bulk Change(see page 120)
- Cloning Multiple Issues(see page 121)
- Using Jira Actions(see page 121)

3.2.5.1 Viewing Issue Details

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

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

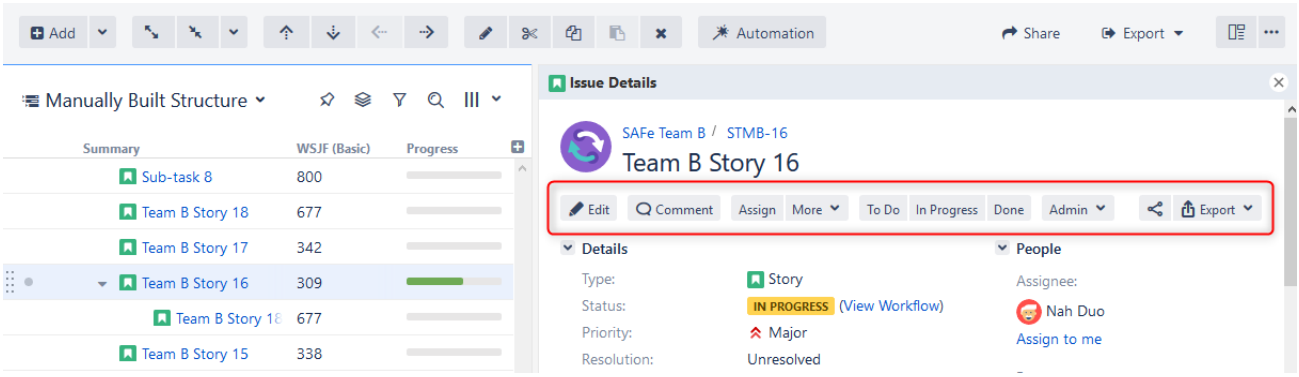


You can define what happens when you click the issue link in a structure. By default, the Issue Details panel is opened. It can be set to open the standard JIRA issue page instead, or do nothing.

To change the default action, go to the **Toggle Panels** menu and select your preference in the **Item Link Action** section.

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[13], view and add comments[14], share[15], view history[16], view development information[17], and more. For specific information on working with and editing issues, please refer to the Jira documentation[18].



To see details for another issue in the structure, simply select another issue(see page 73) 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.

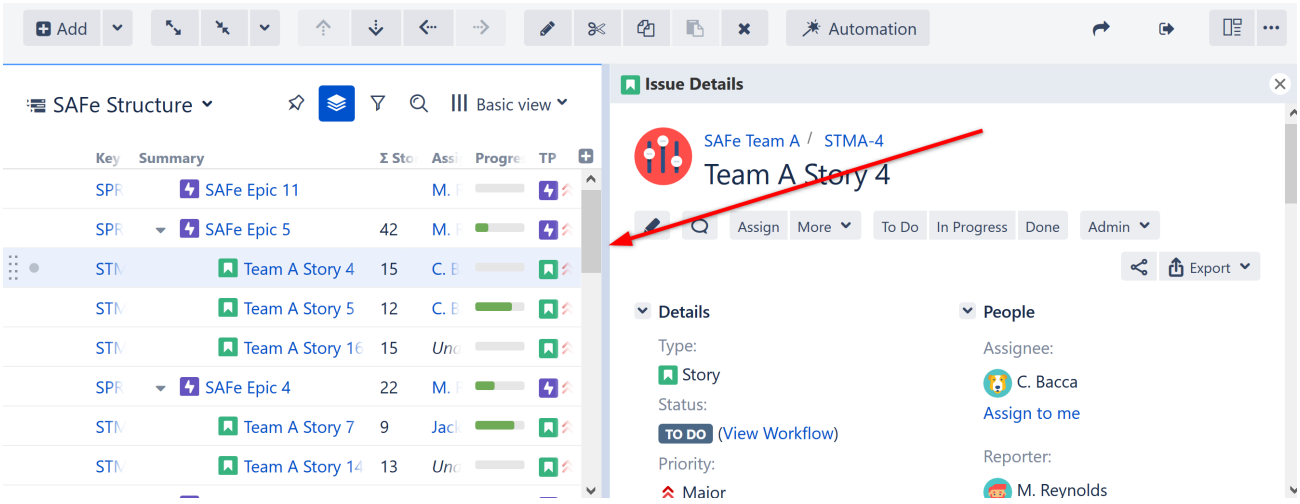
Structure will switch back to your previous view when the details panel is closed.

You can adjust the default views in View Settings(see page 360).

13 https://confluence.atlassian.com/display/JIRA/Editing+an+Issue
14 https://confluence.atlassian.com/display/JIRA/Commenting+on+an+Issue
15 https://confluence.atlassian.com/display/JIRA/Emailing+an+Issue
16 https://confluence.atlassian.com/display/JIRA/Viewing+an+Issue%27s+Change+History
17 https://confluence.atlassian.com/display/JIRA/Viewing+the+Code+Development+Information+for+an+Issue
18 https://confluence.atlassian.com/display/JIRA/Working+with+an+Issue

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.



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 (see page 350)open, when you click an issue in the main panel, the secondary panel(see page 350) 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(see page 350), you can switch keyboard focus between panels using the **** (backslash) shortcut. When the focus is in the Issue Details Panel, keys like ↑ or ↓ (also `PgUp`, `PgDn`, `Home`, `End`, or ↟, ↡, ↖, ↘) scroll the details panel, while all other Structure shortcuts(see page 384) work as usual (including `j` and `k`, which select the next/previous issue in the structure). All shortcuts available to you on the Issue Page should also work as usual: **,** (comma) should open the field selector, **e** should open the Edit Issue dialog, etc.

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

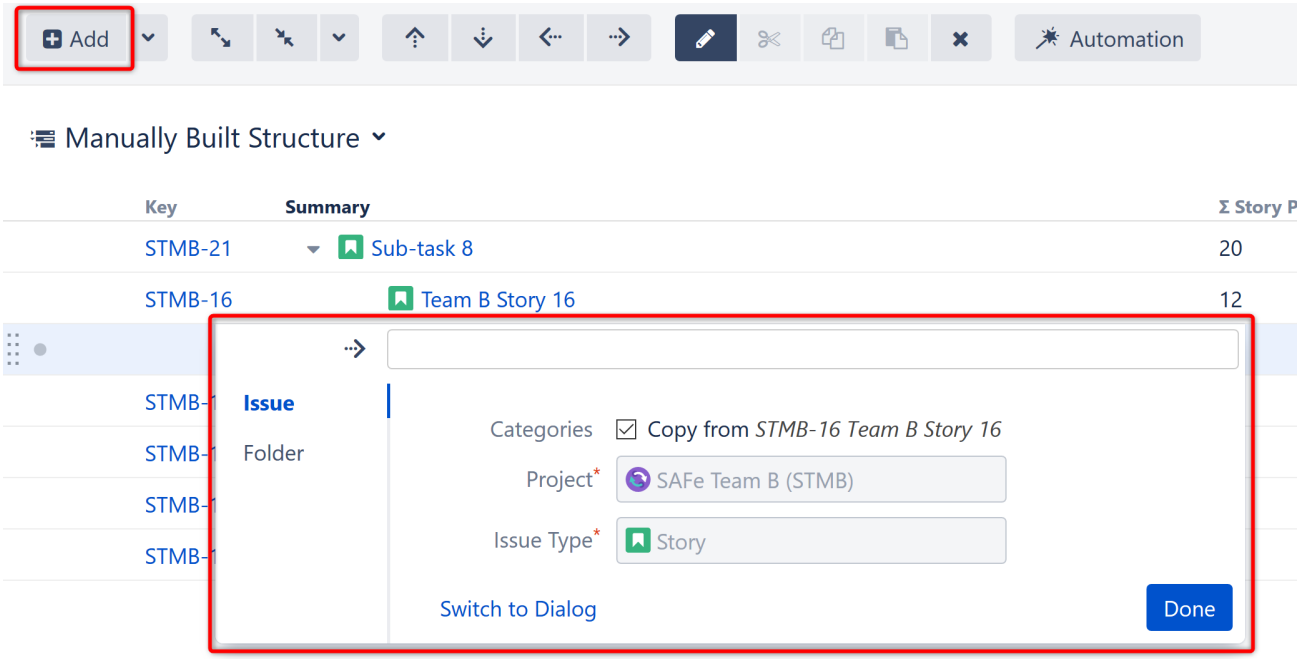
3.2.5.2 Creating New Issues

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

Create a New Issue in Structure

To create a new issue beneath the currently selected item:

1. Use the **Add** button in the toolbar.
2. Enter the Issue Summary in the top entry box of the New Issue panel (to the right of the arrow).
3. To copy attributes (project, issue type, assignee, version information, etc.) from the last selected issue, make sure the **Categories** checkbox is selected. To enter all new information, uncheck this box.
4. Press **Enter** or click **Done** to finish editing and create the new issue on the server.



Keyboard Shortcut

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

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

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

Categories: Copy from...

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

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

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

Editing Other Fields during Creation

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

1. Before you start creating the new issue, add the columns you need to edit to your current view
2. Click **Add** to open the New Issue panel
3. To get to the fields you need to edit, hold **Ctrl+Alt** while using the arrow keys to move between the available columns
4. Once you have finished editing the columns, hold Ctrl+Alt and press the Down arrow to return to the New Issue pane.

You can also edit other fields by clicking the **Switch to Dialog** button, which opens the "Create Issue" dialog(see page 112).

> ⚠ 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.

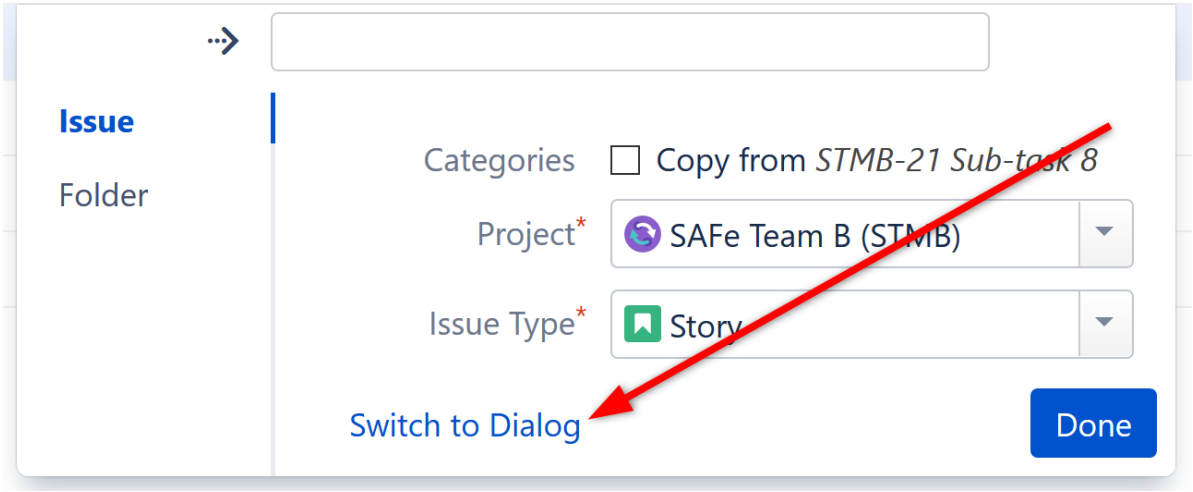
Creating Sub-Tasks

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.

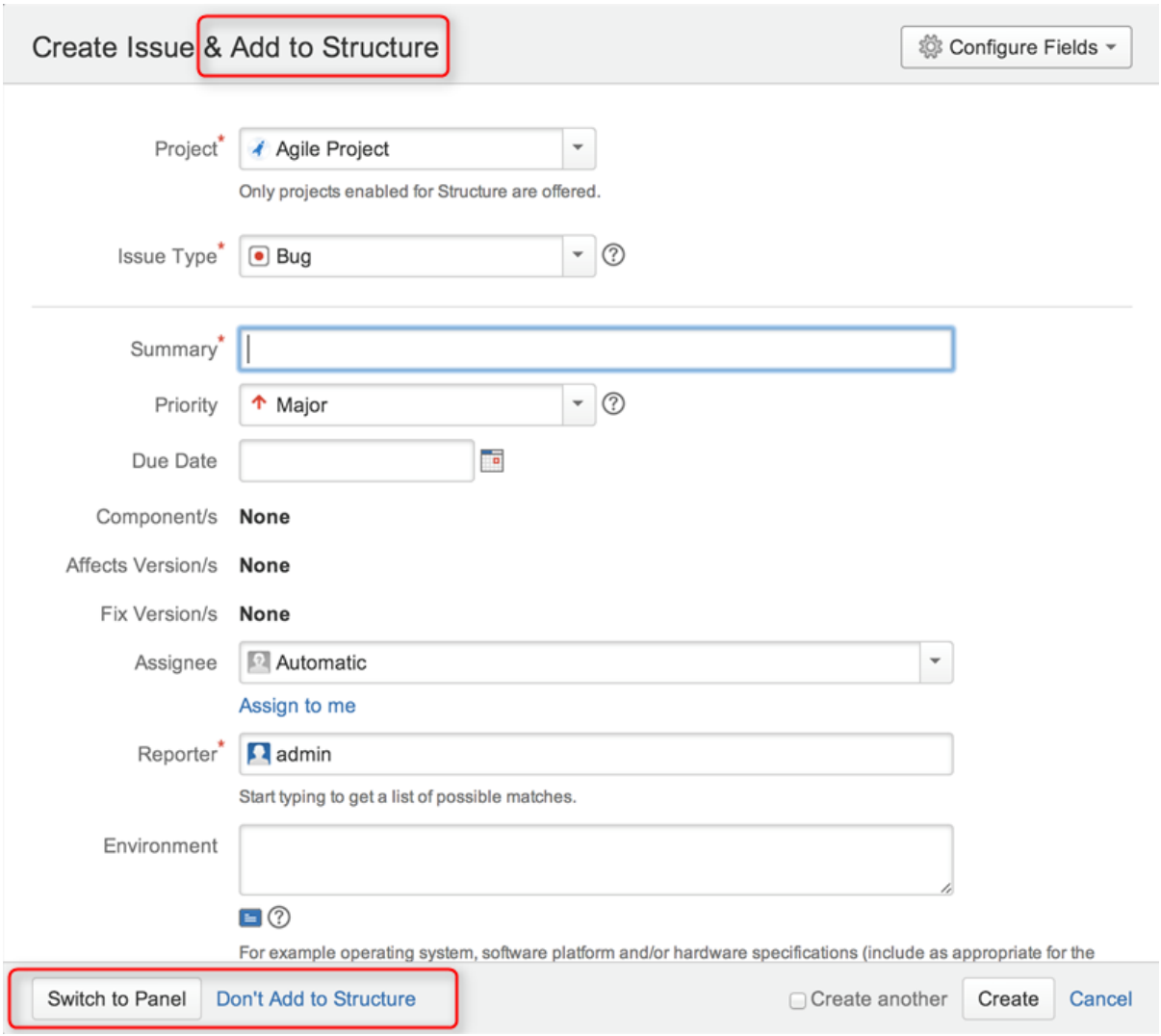
> ⓘ This refers to the Jira sub-task issue type, not Structure sub-issues.

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.



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



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

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

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**:

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

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

Uploading New Issue to the Server

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

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

3.2.5.3 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**

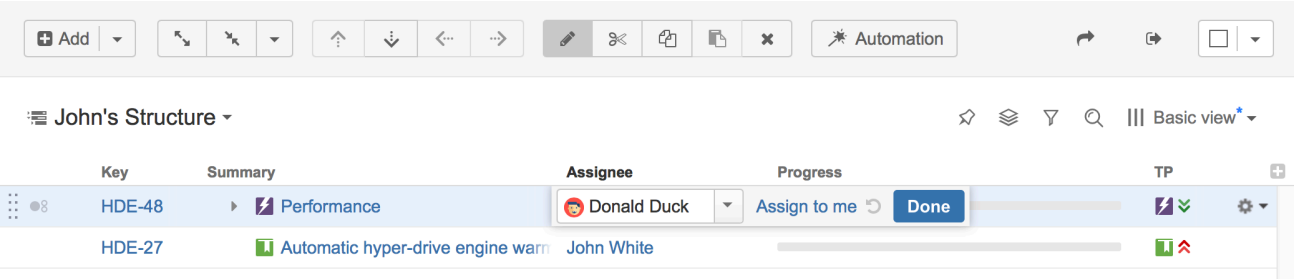
If the value is a link (like in the Summary or Assignee fields), you can still double-click it – the browser will not open the link and will start editing instead.

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(see page 114)).

When working in Edit Mode:

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



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

Learn More

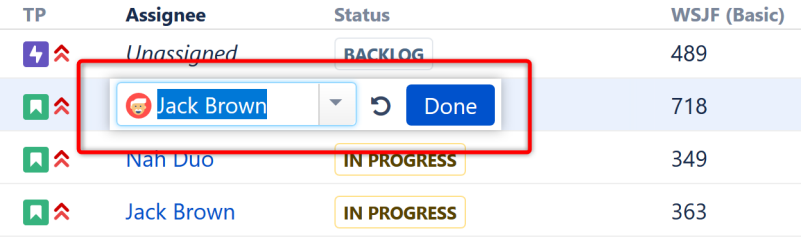
Editing works on every page where Structure widget is displayed; however, there are some limitations when editing issues from the Structure Gadget(see page 119).

- Changing Fields(see page 115)
- Keyboard Shortcuts in Edit Mode(see page 117)
- Correcting Input Errors(see page 119)
- Editing from Gadget(see page 119)
- On E-mail Notifications(see page 120)

> ⚠ In order to edit an issue's fields, you need Edit Issue permission for that issue. If you do not have the correct permission, a read-only flag(see page 317) 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.



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

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

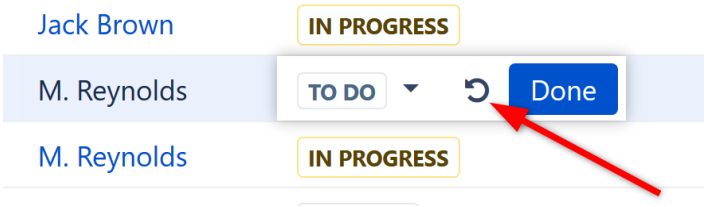
- Click Done
- Click the Edit icon in the toolbar to exit Edit Mode
- Switch to editing another issue, or
- Collapse the hierarchy

> If your Jira is configured to send e-mail notifications about changes, then a notification will be sent as soon as you have finished editing an issue - see On E-mail Notifications(see page 120).

Undoing Changes

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

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

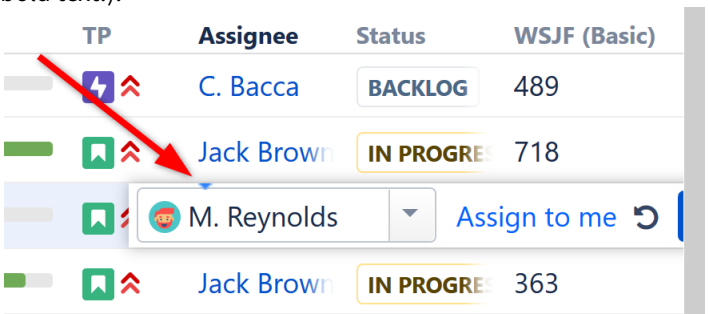


> Hitting **Escape** only reverts the value of the currently edited field. Any other changes will remain.
>
> For example, if you edit an issue's Summary, Assignee and Components fields, but you hit Escape while editing Components, the changes to Summary and Assignee will still be uploaded.

The Field Editor

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

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



Allowed Changes

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

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

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

Keyboard Shortcuts in Edit Mode

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

Entering Edit Mode

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

Keyboard Shortcuts in the Edit Mode

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

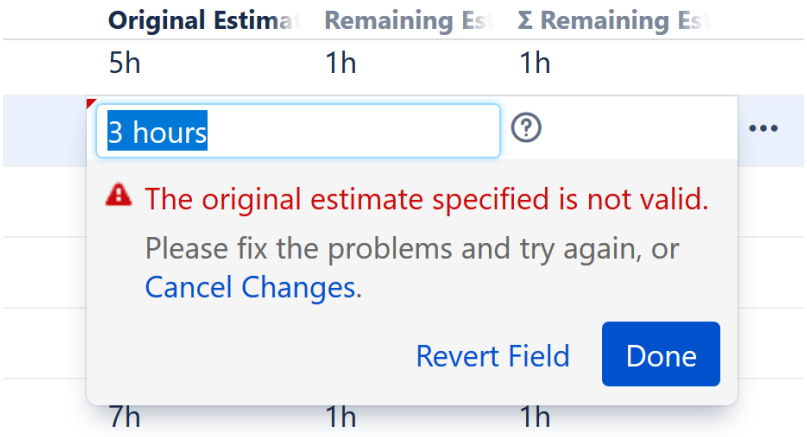
Keyboard Shortcut	Action
Ctrl+Alt+↓	Edit the same field of the next editable issue.
Ctrl+Alt+↑	Edit the same field of the previous editable issue.
Ctrl+Alt+→	Edit next editable field. Unlike **Tab**, this combination will not move editing to the next issue.
Ctrl+Alt+←	Edit previous editable field. Unlike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

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

See Also: Keyboard Shortcuts(see page 384)

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

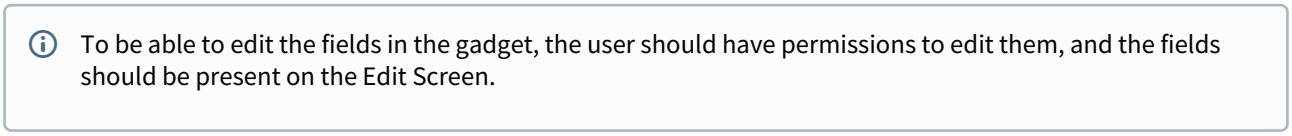
The Structure Dashboard Gadget allows editing issues too, but due to some incompatibilities between field editors and gadget framework, not all fields can be edited.

Structure Notes(see page 319) and the following Jira fields are editable from Structure Gadget:

- Summary

- Assignee
- Issue Type
- Priority
- Reporter
- Security Level
- Original Estimate
- Remaining Estimate
- Numeric Custom fields (like Storypoints)

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


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

The Confluence gadget is not editable.

On E-mail Notifications

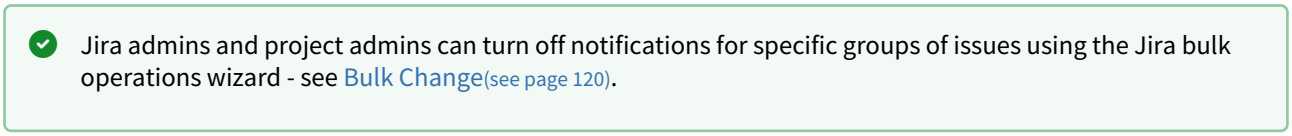
When issues are updated in Structure, notifications are sent based on the corresponding project's notification settings in Jira. This usually means everyone involved with the issue will receive an email notification.

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

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

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

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


Jira admins and project admins can turn off notifications for specific groups of issues using the Jira bulk operations wizard - see Bulk Change(see page 120).

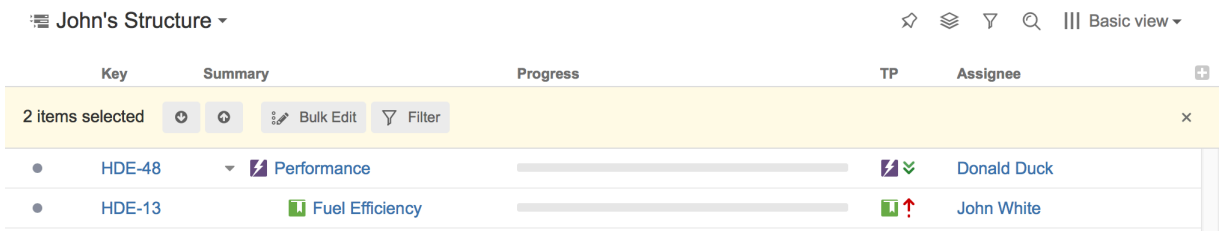
3.2.5.4 Bulk Change

With Structure, you can quickly select multiple issues(see page 101) 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(see page 384) for selecting issues.

2. Click **Bulk Edit**.



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

3.2.5.5 Cloning Multiple Issues

Structure allows you to copy the whole structure and clone all issues in the structure. See Copying Structure and Cloning Issues(see page 368).

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

1. Select issues you'd like to clone using multiple selection(see page 101).
2. Use **Copy** action on the toolbar (or hit Ctrl+C / Command+C) to copy the issues to the Issue Clipboard(see page 103).
3. Use **Structure | Create Structure** menu and create a new temporary structure, let's call it **T1**.
4. Open the new structure and use **Paste** action to add issues from clipboard.
5. Copy and clone structure **T1** – see Copying Structure and Cloning Issues(see page 368). Let's name the resulting copy **T2**.
6. Open **T2**, select all issues (use Ctrl+A / Command+A).
7. Use issue clipboard in the same way to copy cloned issues back to the structure where they are needed.
8. Delete structures **T1** and **T2**.

3.2.5.6 Using Jira Actions

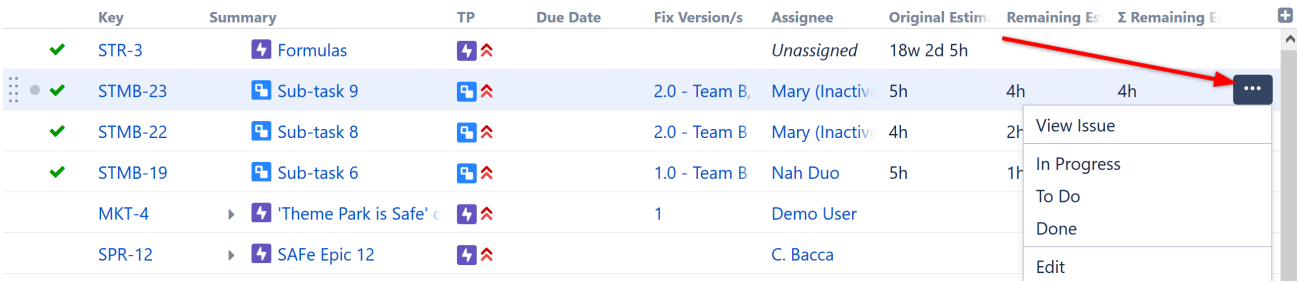
Jira actions can be applied to issues directly from Structure, using the Jira Actions drop-down and Jira keyboard shortcuts.

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.



The Jira Actions menu is unavailable when the Issue Details panel (see page 108)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(see page 85), 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(see page 83), most actions do not require page a reload.

3.3 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.

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

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

3.3.1 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.

3.3.2 Types of Generators

- Insert Generators(see page 125) — Insert generators allow you to automatically add issues to the structure.
- Filter Generators(see page 130) — Filter generators allow you to limit the scope of your structure by removing any issues that do not pass certain criteria.
- Sort Generators(see page 139) — Sort generators allow you to order your structure based on a Jira attribute, Structure attribute, or Agile rank.
- Group Generators(see page 141) — Group generators allow you to group issues by most standard Jira fields, custom fields provided by Jira and other issue attributes.
- Extend Generators(see page 146) — Extend generators allow you to add issues to a structure based on Issue Links, Epic Links and Sub-task relationships.

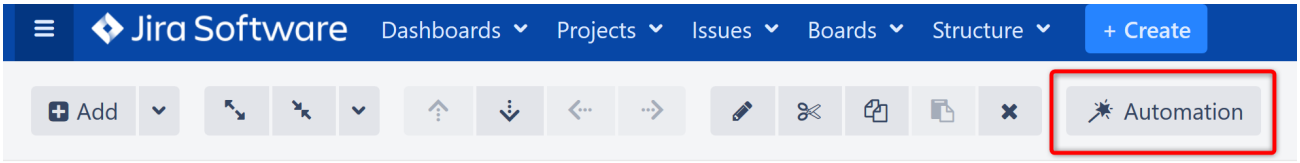
3.3.3 Generator Scope

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.

3.3.4 Adding a Generator

3.3.4.1 Automation Editing Mode

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



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

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

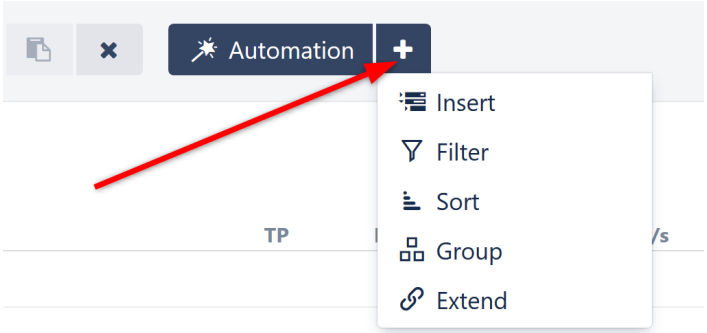
3.3.4.2 Adding a Generator

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

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

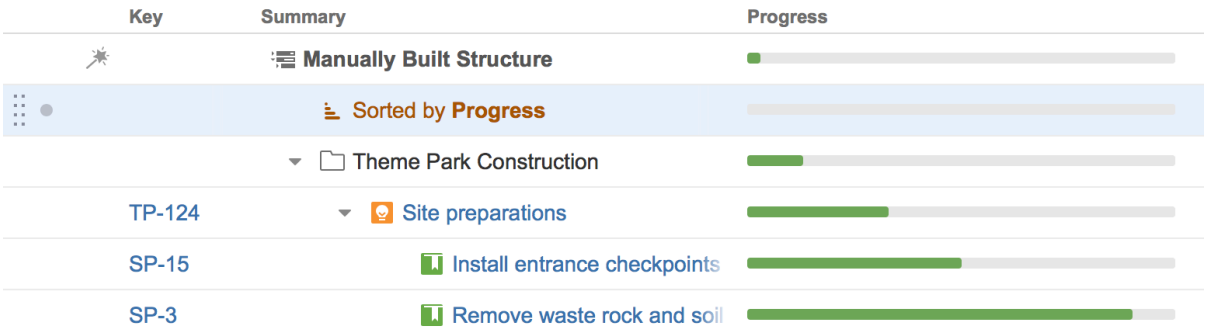
> You can only add generators under static parts of the the structure. You cannot add them under dynamic items (items added by other generators).

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(see page 124).

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



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

3.3.5 Types of Generators

There are several types of generators available by default:

- Insert Generators(see page 125) — Insert generators allow you to automatically add issues to the structure.

- Filter Generators(see page 130) — Filter generators allow you to limit the scope of your structure by removing any issues that do not pass certain criteria.
- Sort Generators(see page 139) — Sort generators allow you to order your structure based on a Jira attribute, Structure attribute, or Agile rank.
- Group Generators(see page 141) — Group generators allow you to group issues by most standard Jira fields, custom fields provided by Jira and other issue attributes.
- Extend Generators(see page 146) — Extend generators allow you to add issues to a structure based on Issue Links, Epic Links and Sub-task relationships.

3.3.5.1 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(see page 146) to add stories and sub-tasks)
- Use a JQL query to add all issues assigned to you (and then organize them with a Group Generator(see page 141) or Sort Generator(see page 139))
- Insert issues from specific projects into their own folders within the structure
- Compile multiple structures into a single structure

> Insert generators can only add issues from Structure-enabled projects(see page 477).

Types of Inserters

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

- Agile Board Inserter(see page 125) — Adds issues from a selected Agile board into the structure.
- JQL Query Inserter(see page 127) — Adds the results of the JQL query to the structure.
- Structure Inserter(see page 128) — Adds a selected structure into the current structure.
- Text Query Inserter(see page 129) — Adds the results of a text search to the structure.

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

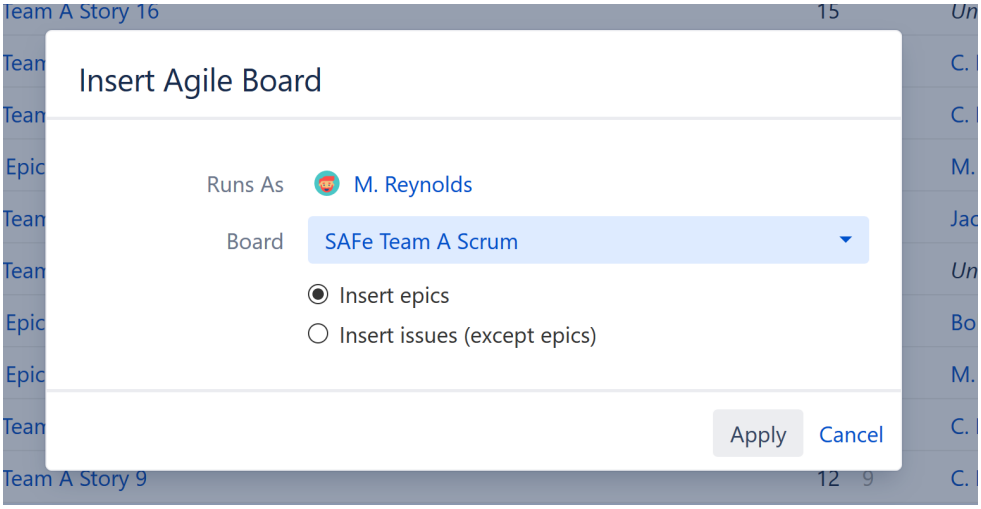
Always Up-To-Date

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.

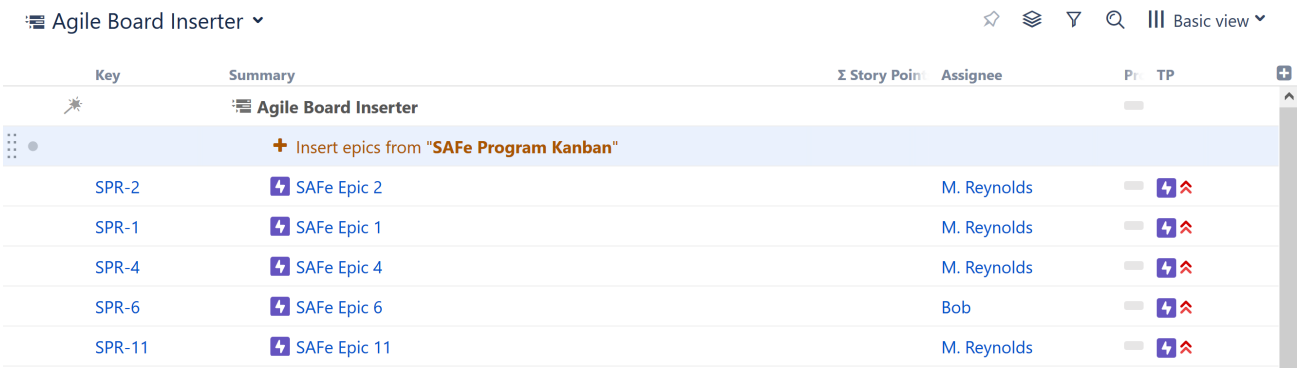
Agile Board Inserter

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(see page 146) or Group(see page 141) generators later.

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



If you want to include items from more than one Agile Board, you can add an additional Agile Board Inserter – or use the JQL Inserter,(see page 127) which allows you to specify multiple projects using the 'OR' keyword.

When to Use the Agile Board Inserter

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

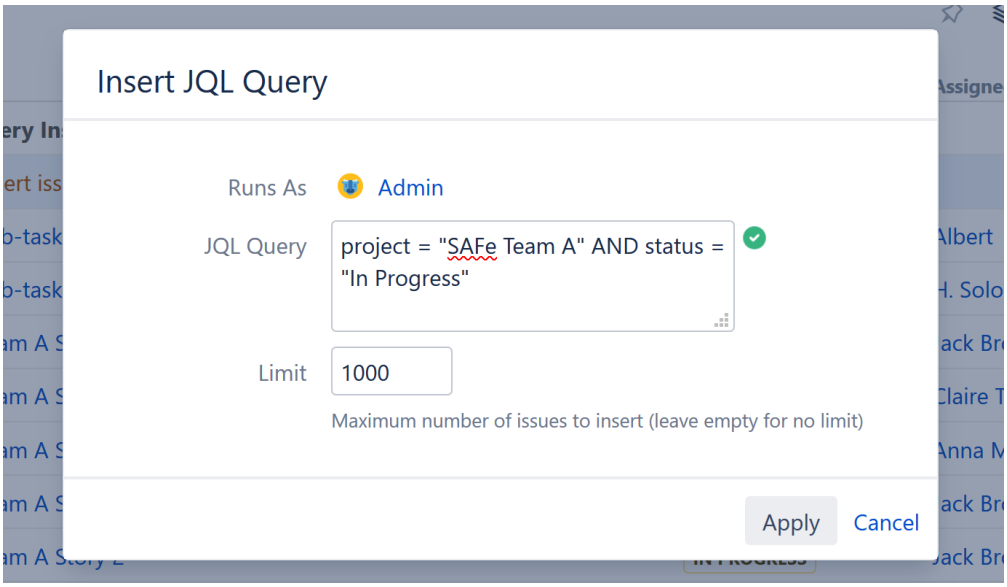
Here's a quick example of what that might look like:

1. First, add all the epics from a specific Agile Board.
2. Next, use the Extend (see page 146)generator to add stories beneath each epic.
3. Then use another Extend generator to add linked issues under those stories. The resulting hierarchy will allow you to quickly review stories and their dependencies.
4. Finally, you can add Sort (see page 139)or Filter(see page 130) generators to tailor the results to your specific needs – or use Transformations(see page 184) to temporarily sort or filter your results.

You can expand and customize Inserter results using the Extend(see page 146), Filter(see page 130), Sort(see page 139) and/or Group(see page 141) generators.

JQL Query Inserter

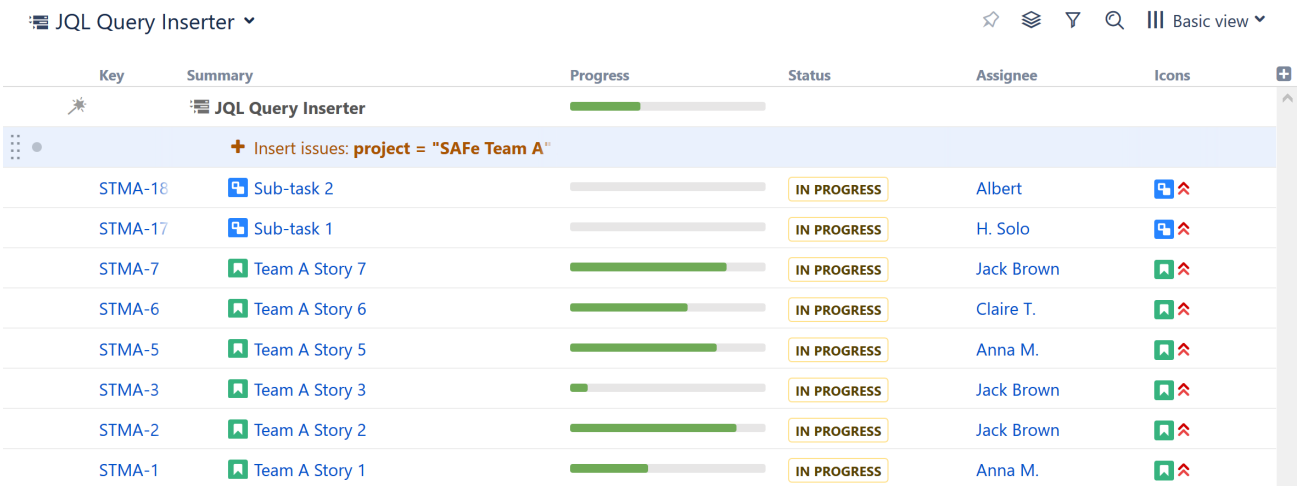
The JQL Query Inserter allows you to add issues based on a JQL query.



To determine the scope of issues that should be added to the structure, simply add the appropriate JQL query[19].

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.



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

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.

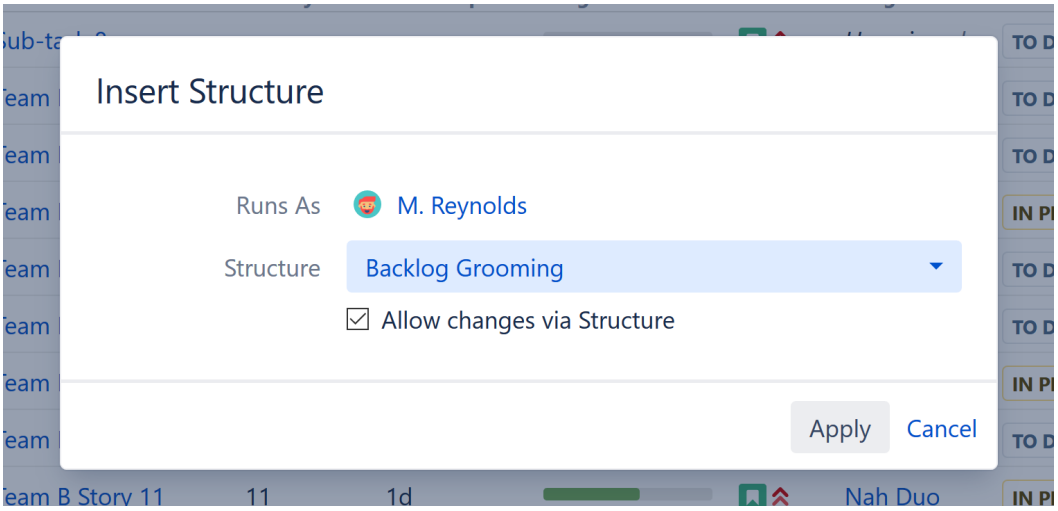
Here are a few examples of when you might want to use the JQL Inserter:

- 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(see page 146), Filter(see page 130), Sort(see page 139) and Group(see page 141) 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.

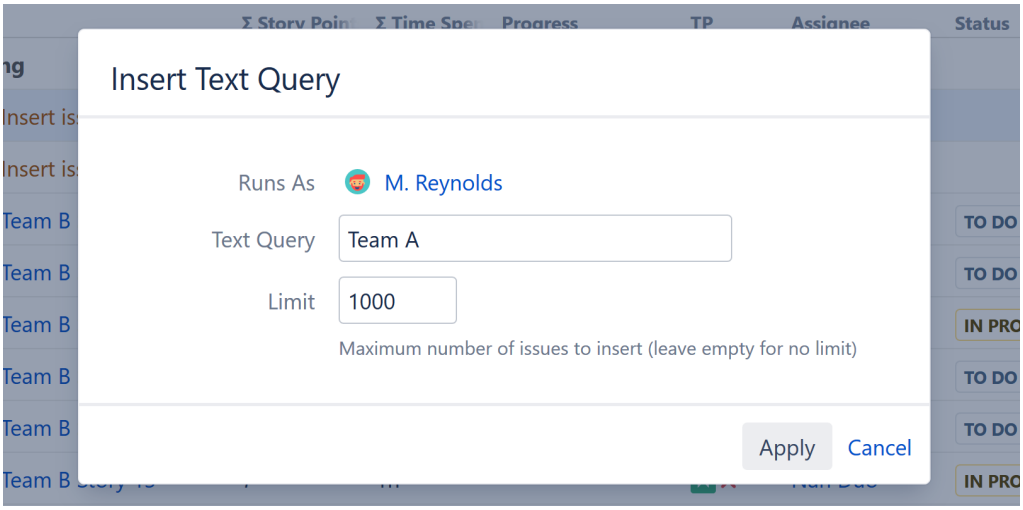
When to Use the Structure Inserter

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

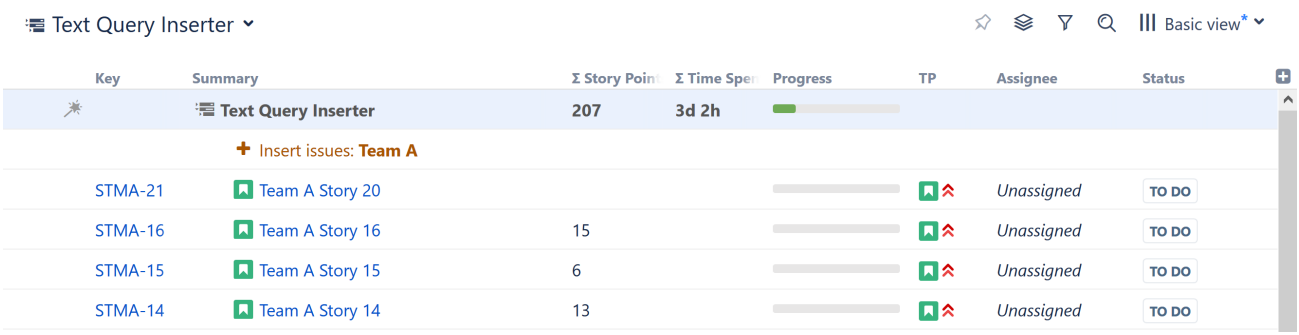
The Text Query Inserter allows you to add issues based on the text contained within their summaries.



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.



When to Use the Text Query Inserter

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

For example:

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

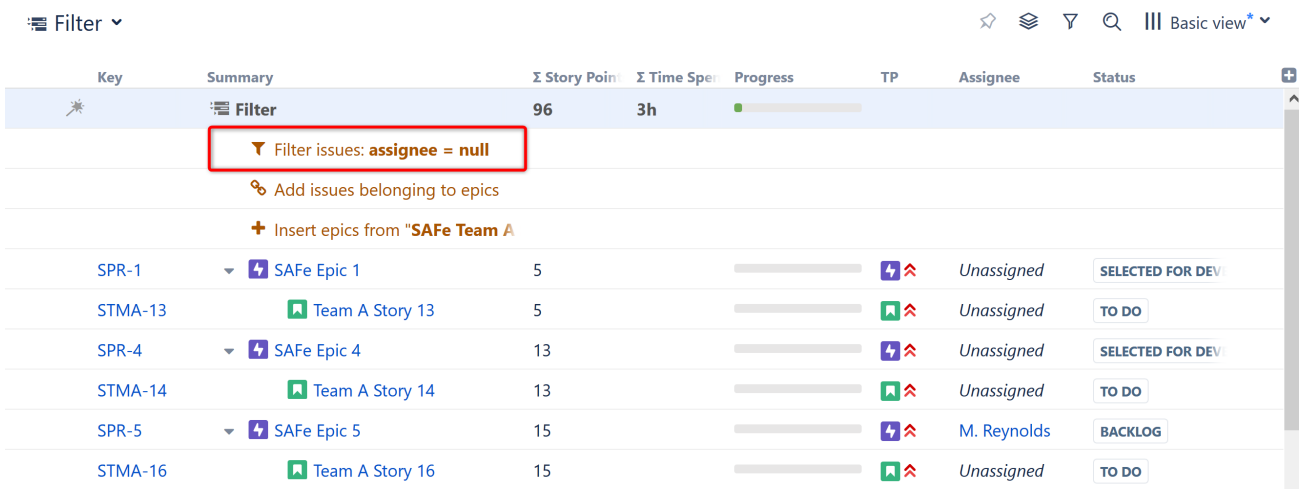
You can expand and customize your results using the Extend(see page 146), Filter(see page 130), Sort(see page 139) and/or Group(see page 141) generators.

3.3.5.2 Filter Generators

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

- Hide Closed Sprints
- Remove Inserter/Extender Duplicates(see page 133)
- JQL Query
- S-JQL(see page 266) Query
- Sprints - filter by active, future, or past sprints
- Text Query

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



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

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

When to Use the Filter Generator

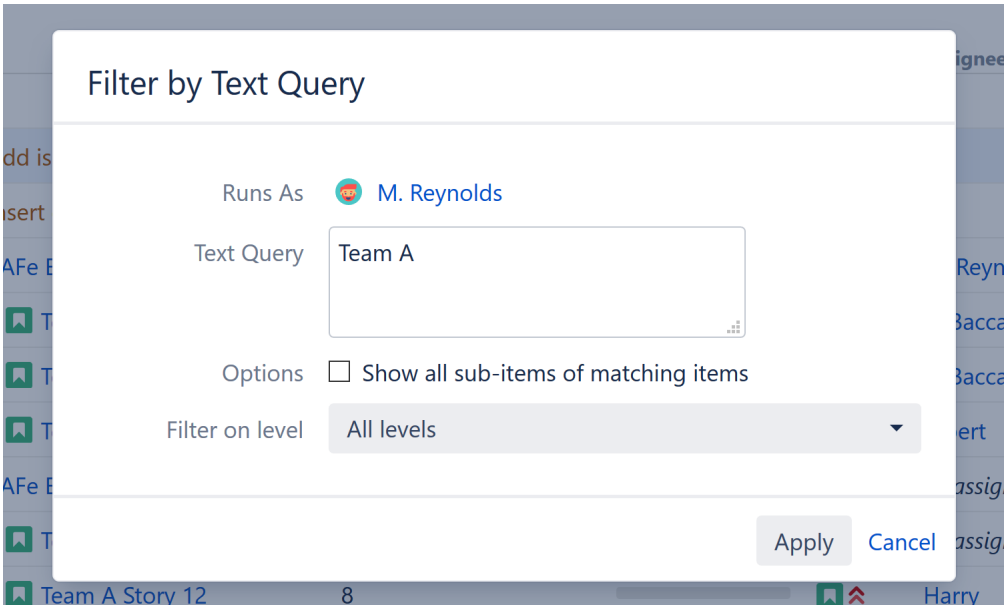
The Filter generator can be used with a manually-built structure, or combined with an Insert generator(see page 125) to limit the issues that are added.

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

Use the Filter generator when you always want issues filtered within the structure. To temporarily filter issues, use the Filter(see page 183) button in the toolbar or Quick Transformations(see page 188).

Configuring a Filter

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

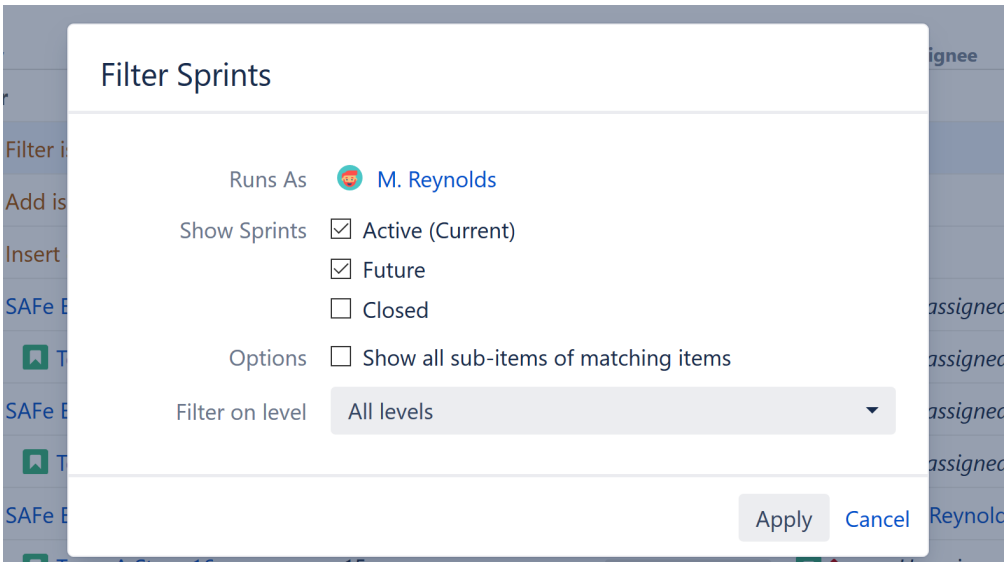


While each filter has its own options, they all include the following:

- **Show all sub-items of matching items** - If this option is selected, all issues that match your filter criteria will be included in the structure, along with any sub-items of those issues.
- **Filter on level** - You can apply a filter to specific levels within your hierarchy. For example, you may want to include all top-level items, but then filter the stories beneath them. See Generator Scope(see page 169) 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.

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.

Inserter/Extender Duplicates Filter

When you build a structure using both Insert(see page 125) and Extend(see page 146) generators, there is a chance you will wind up with duplicate issues – the Inserter/Extender Duplicate filter allows you to quickly remove those duplicate issues from your structure.

Here's how this can happen:

1. You use an Insert generator to add every Story from a current project. Those stories are added to the the top level of your structure.
2. You then use the Linked Issues Extender (see page 147)to add issues linked to those stories - these issues will be placed below your existing stories in the structure.
3. If any of the original stories are linked to each other, they will appear more than once in your structure – at the top level (because they were added by the inserter) and as children of other issues (because they were added again by the extender).

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

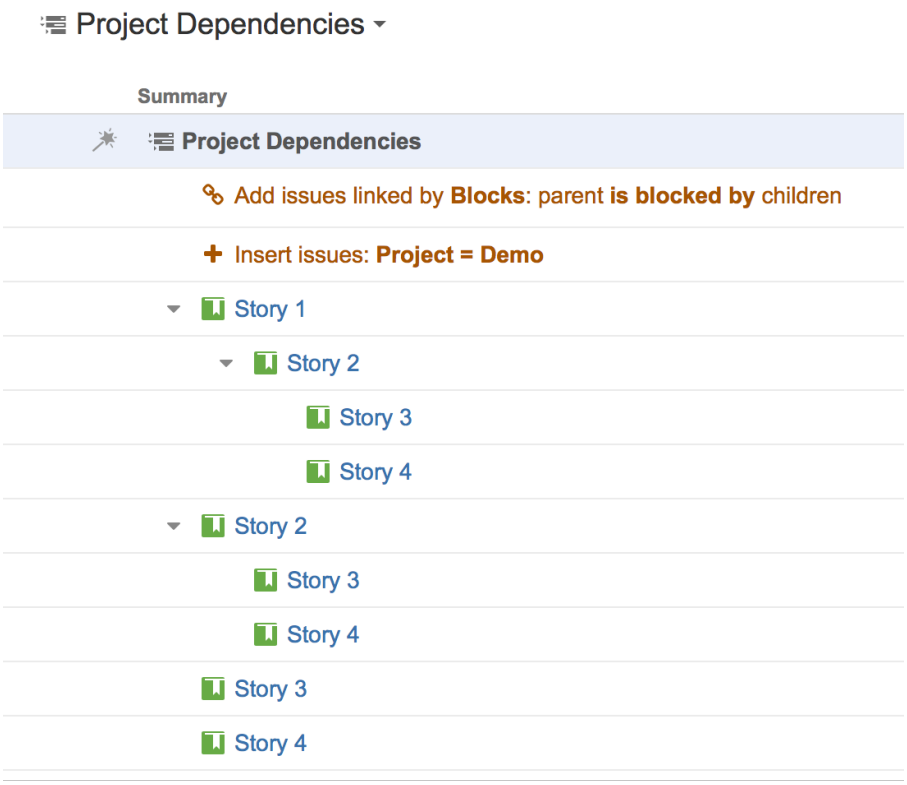
Examples

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

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

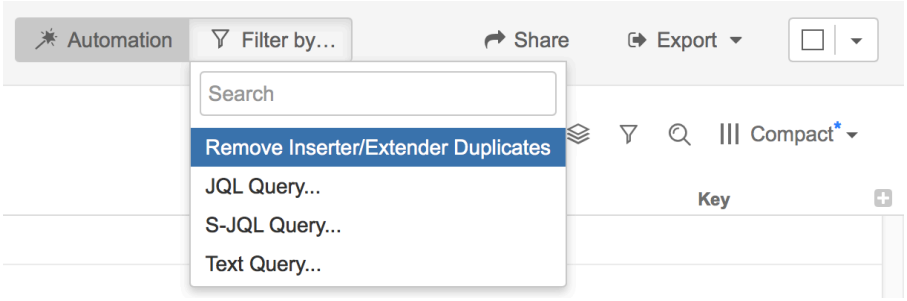
In our structure, we want to see all issues from our project arranged based on the existing "Blocks" links.

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

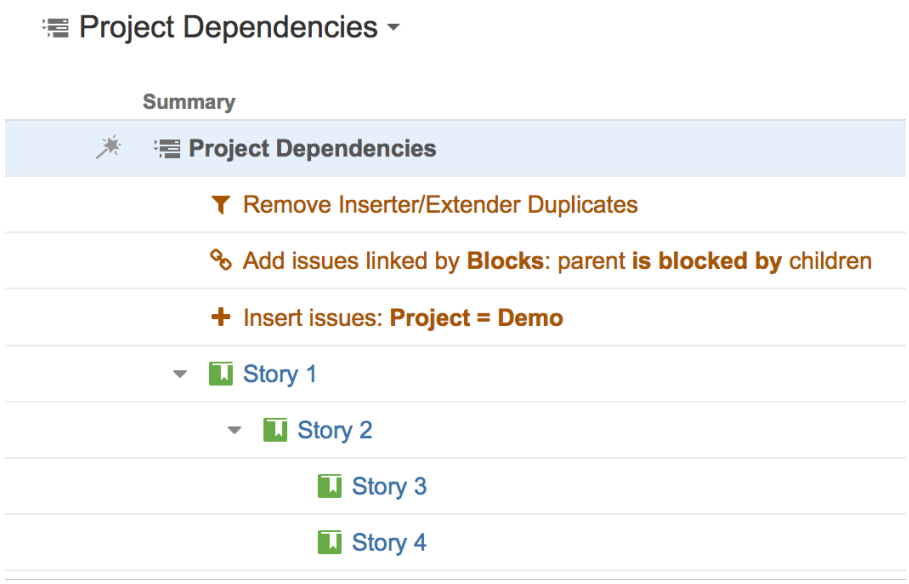


You can see that some issues have been duplicated, because the Extender adds the children under parents, even if they are already in the structure.

Now let's add the Duplicates Filter:



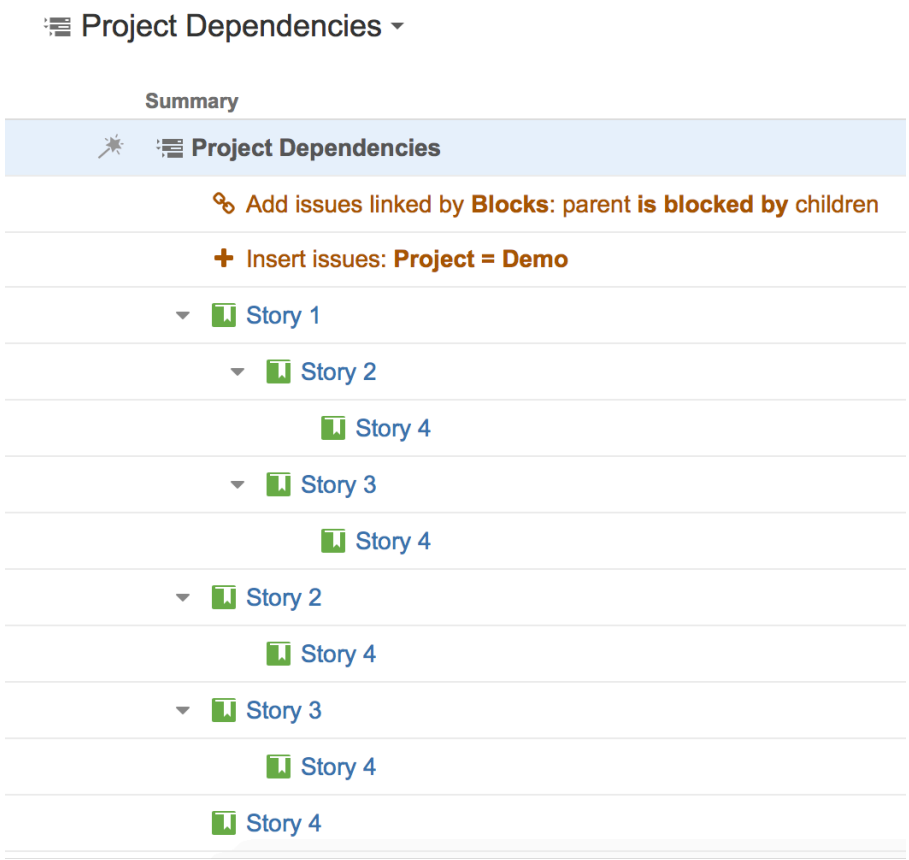
As a result, we get a structure with the hierarchy and no duplicates:



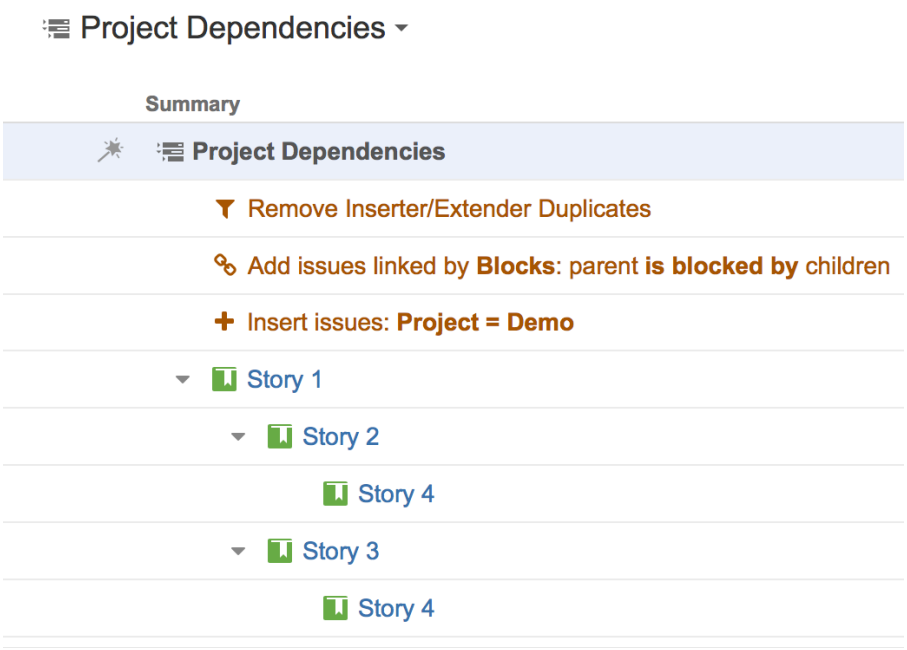
Multiple Parents
We have the same situation as in the example above, but we have one story that blocks two other stories, so it should be shown under both of them:

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

Without Duplicate Filter, it looks like this:



With the filter applied, any identical instances are removed:

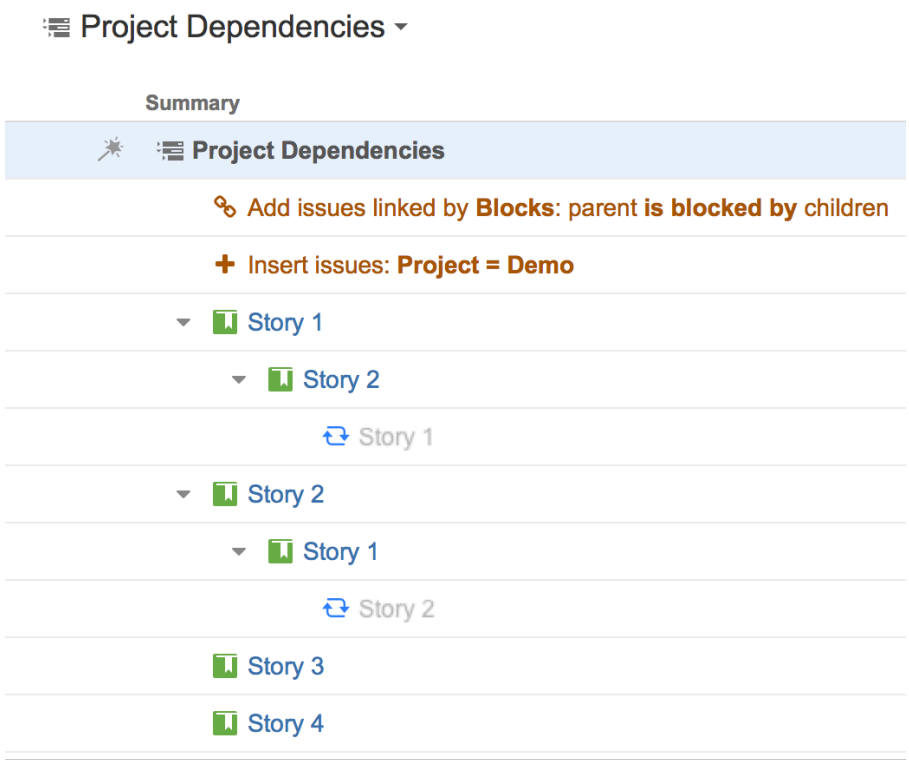


In this example, Story 4 still appears twice – this is because these two instances are not identical. One is blocking Story 2, while the other is blocking Story 3. Both need to be expressed in the hierarchy.

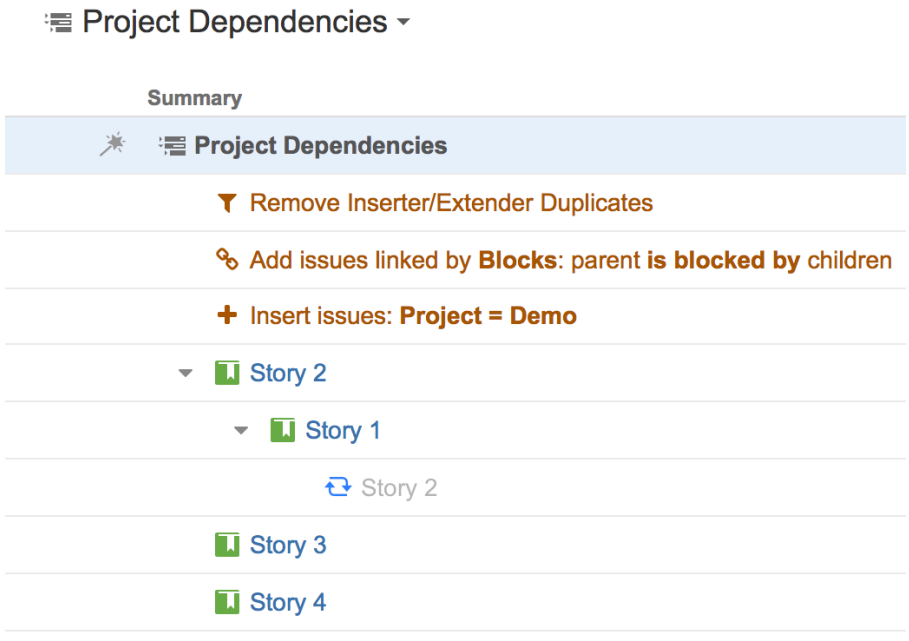
Link Cycles
If there are link cycles between the issues (both issues link to each other), the Duplicates Filter will remove one of the branches and keep the other, to make sure all the issues added by the Insert and Extend generators are in the structure.

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

Without the filter, we get the following structure:



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



3.3.5.3 Sort Generators

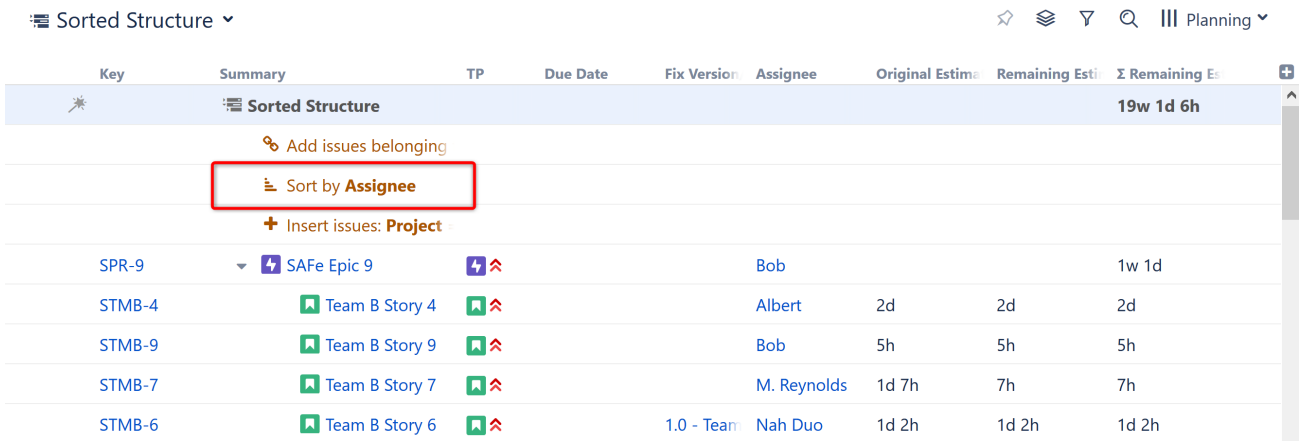
Sort generators allow you to order your structure based on a Jira attribute, Structure attribute, or Agile rank. They can also be used to allow manual sorting for JQL-generated structures.

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.

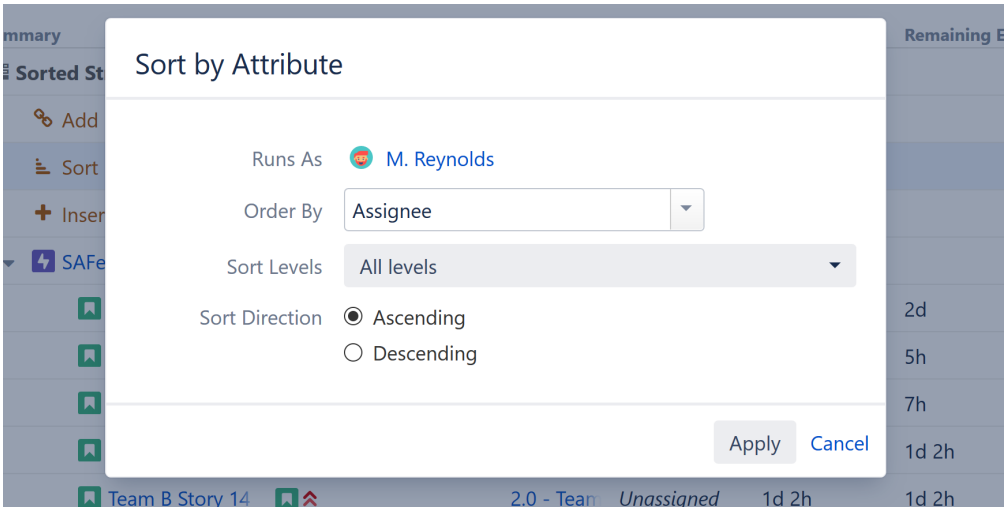
> ⚠ Placement matters. Generators only affect issues beneath them, so if you want to sort the entire structure, place the generator at the very top (by selecting the structure's name in the top row). If you place it anywhere else, it will only sort the items beneath it.

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.



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 (see page 125)issues from all active projects (and use Extenders(see page 146) as appropriate), and then apply the following sorts:

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

See Generator Scope(see page 169) 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.

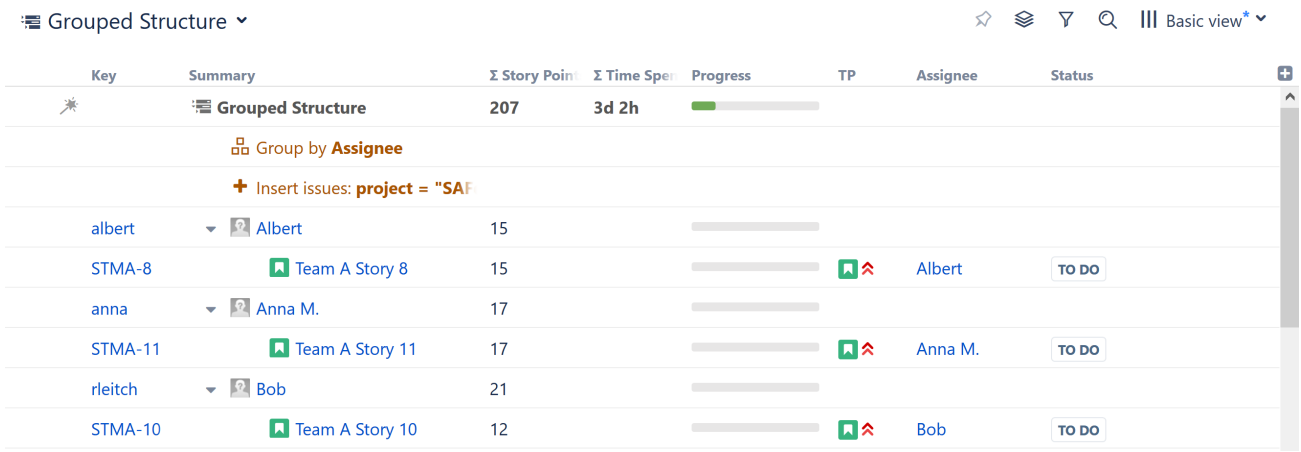
Manual Reordering Generator

If your structure was built using a JQL Query Inserter(see page 127), by default you will not be able to reorder issues within the structure. The **Manual Reordering Generator** allows you to move issues up or down, provided they remain at the same level within the hierarchy.

To move items between levels, you can enable Manual Adjustments(see page 175).

3.3.5.4 Group Generators

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



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

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

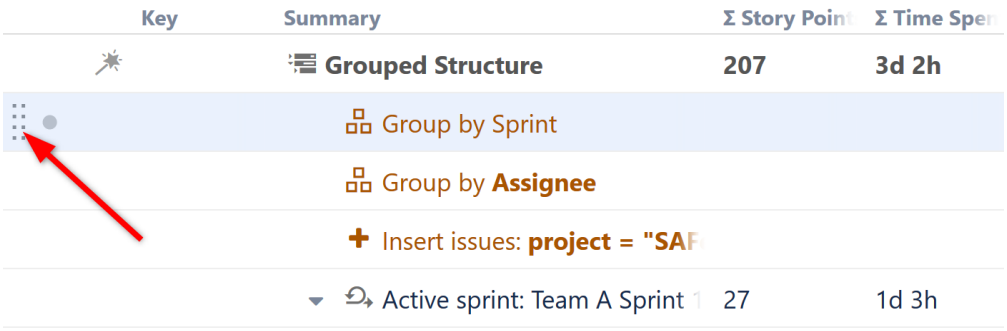
> ⚠ Placement matters. Generators only affect issues beneath them, so if you want to group the entire structure, place the generator at the very top (by selecting the structure's name in the top row). If you place it anywhere else, it will only group the items beneath it.

Grouping Order

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

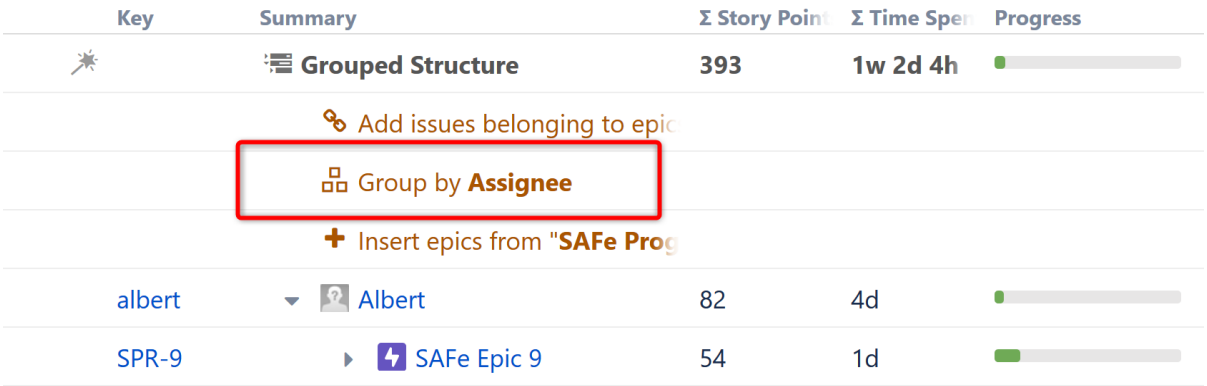
In the example below, we used an Insert generator to add issues, and then added Group by Sprint and Group by Assignee generators. Two new levels were created, grouping our issues first by Sprints and then by Assignees:

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

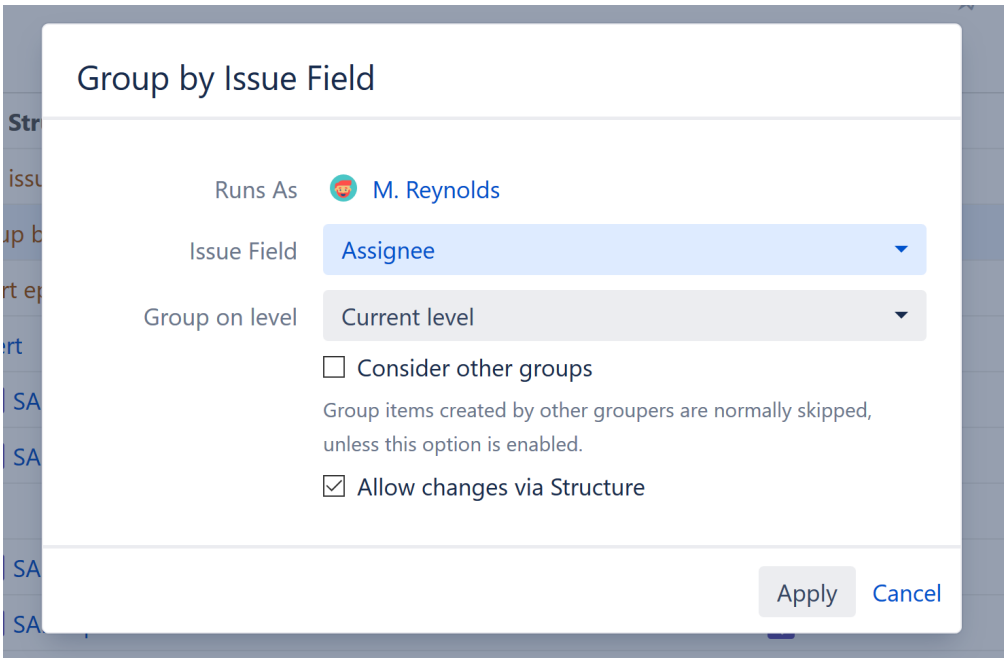


Customize Your Grouping

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



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



- **Group on level** - You can apply the grouping to the current level, next level, or manually enter a level within the hierarchy. See Generator Scope(see page 169) 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(see page 125)).
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
- **Advanced Roadmaps parent link:** as defined in Advanced Roadmaps for Jira(formerly Portfolio for Jira)
- **Tempo Account:** as defined in Tempo for Jira
- **Issue links**: group issues by their linked issues. With this generator, you can select link type and direction. For example, you can group issues under their respective blockers (issues that block them).
- **Customer Request Type**: as defined in Jira Service Management (formerly Jira Service Desk)

> 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.

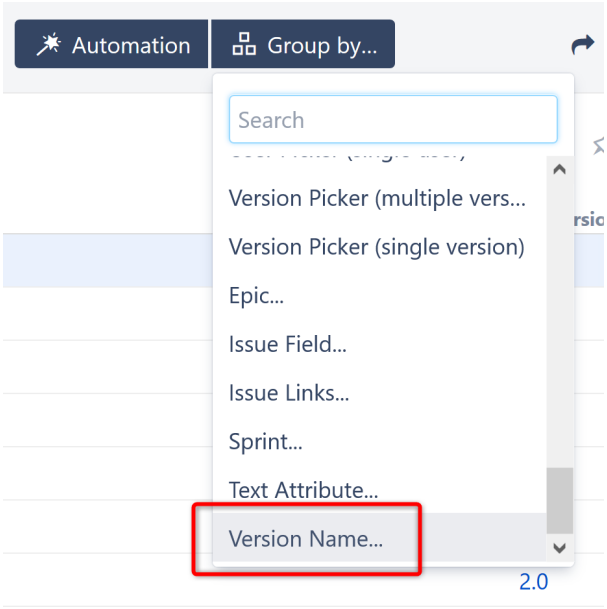
Additionally, if the **Allow changes via Structure** option is enabled, the following will happen when you move an issue between these groups:

- **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

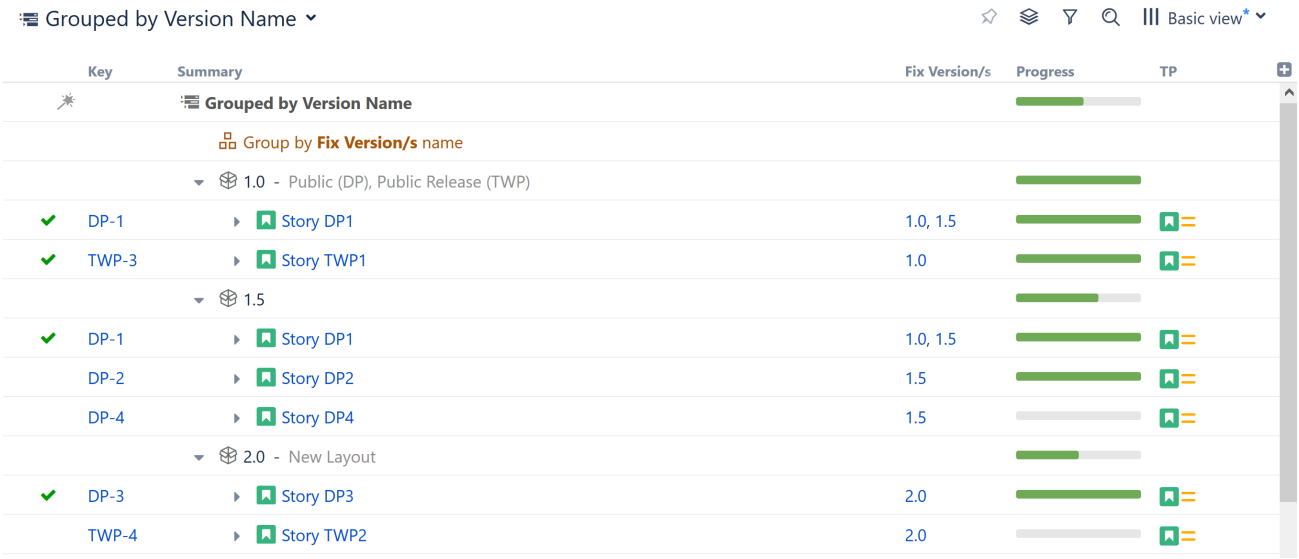
Using the **Version Name...** Grouper, you can group issues with the same version names across multiple projects.



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.



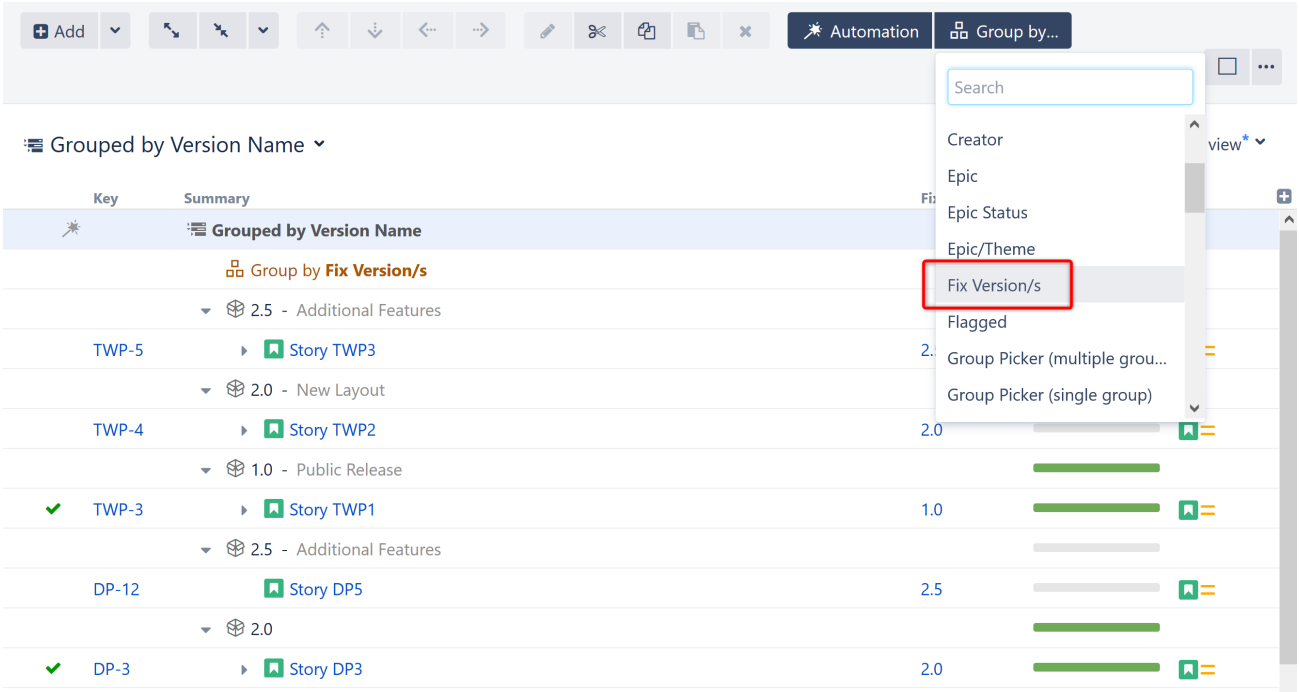
The Version Name Grouper is case insensitive and ignores spaces, so "Version 1" and " version1" will be in the same group.

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

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

Grouping by the Attribute Rather Than Version Name

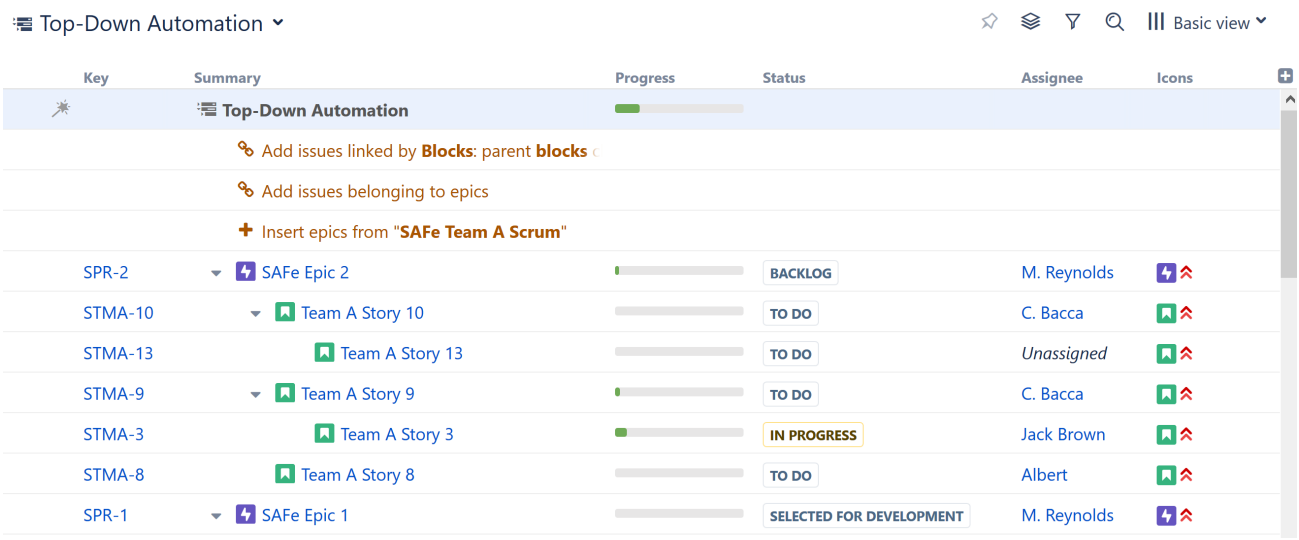
The **Version Name...** Grouper creates a single group for each version, regardless of which project the issues come from. If you prefer to have separate groups for separate projects, instead of selecting **Version Name...** from the Group by menu, select the specific version attribute you wish to group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



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

3.3.5.5 Extend Generators

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



Extend generators are often used in conjunction with an Insert generator(see page 125). In the example above, we used the Agile Board Inserter(see page 125) to add our epics, followed by the Stories Under Epics Extender(see page 149) to add the stories beneath each epic and, finally, the Linked Issues Extender(see page 147) to add linked issues beneath these stories.

Types of Extenders

The following Insert generators are available:

- Linked Issues Extender(see page 147) — Pulls in issues that are linked to issues already in the structure.
- Stories Under Epics Extender(see page 149) — Pulls in issues belonging to epics already in your structure.
- Sub-tasks Extender(see page 150) — Pulls in sub-tasks belonging to issues already in your structure.
- Child Issues for Portfolio Extender(see page 151) — Pulls in child issues using the Parent Link field from the Portfolio for JIRA add-on (if installed).
- Pages Extenders(see page 152) — Pulls in child pages or linked pages from Confluence (when the Structure.Pages add-on is installed).

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

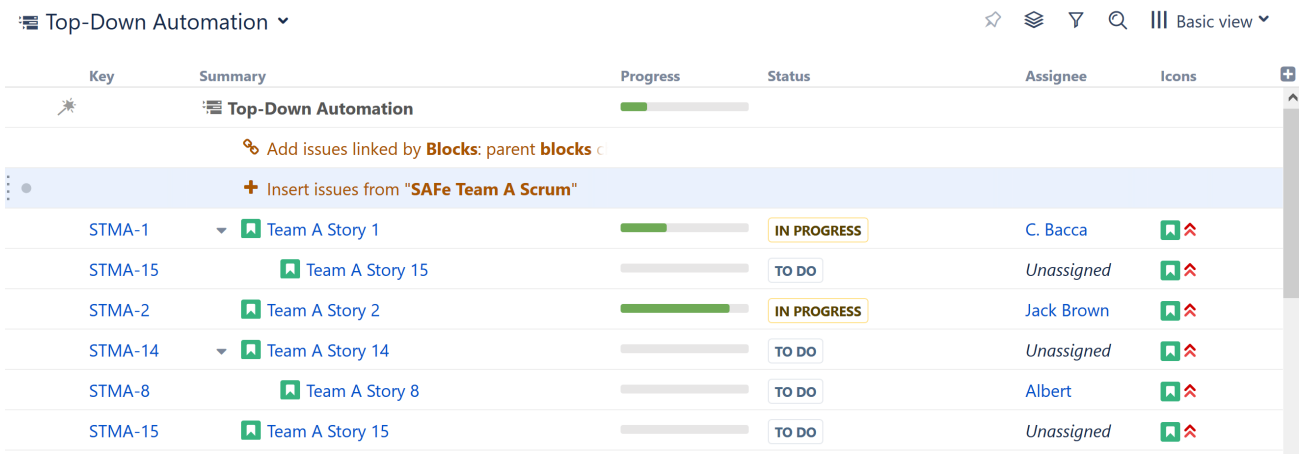
Always Up-To-Date

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.

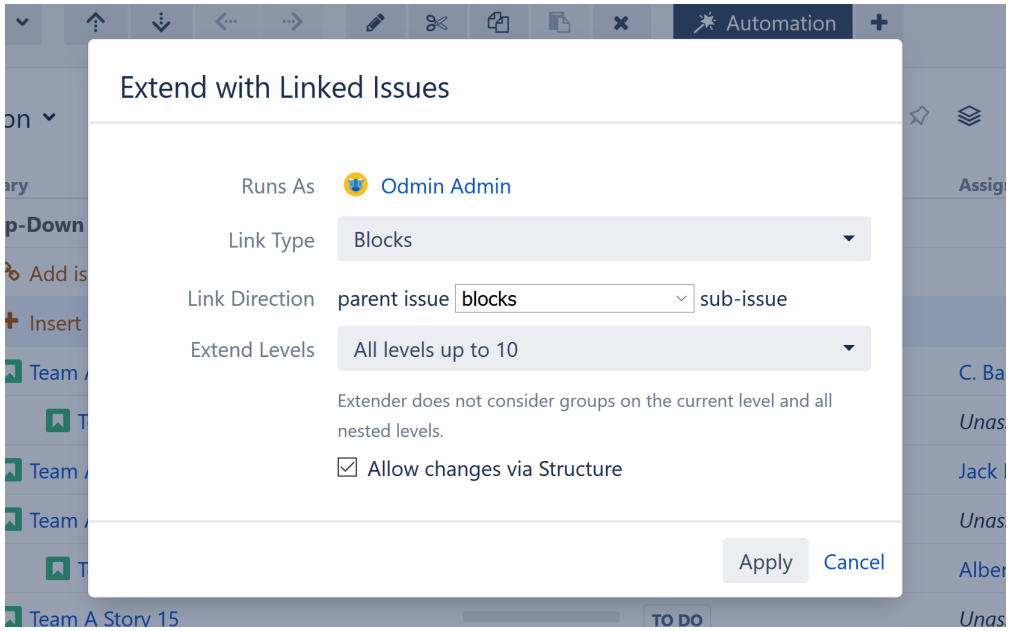


When using the Linked Issues Extender, some issues may appear in your structure more than once. In the example above, "Team A Story 15" appears twice, because it met the criteria for the original Inserter AND it was linked to "Team A Story 1."

You can use the Inserter/Extender Duplicates Filter(see page 133) to remove these duplicates from your structure.

Customize Your Extender

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



You can customize:

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

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

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

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

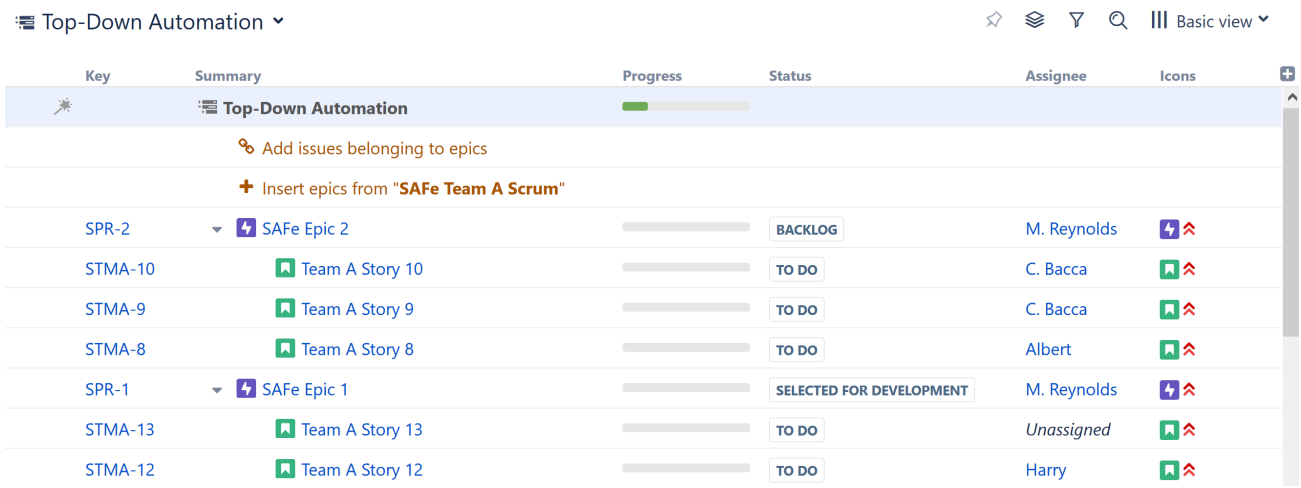
See Generator Scope(see page 169) to learn more about customizing levels.

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

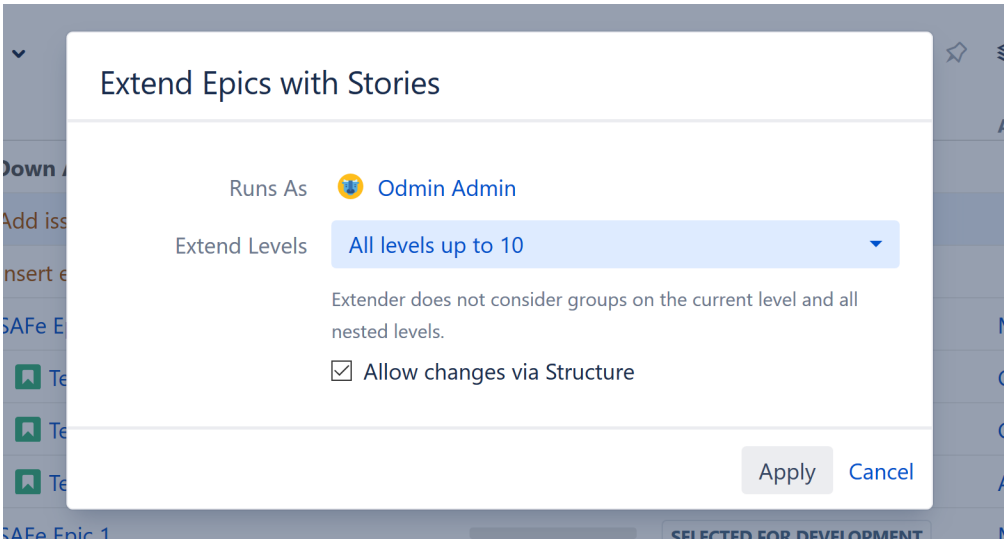
The Stories Under Epics Extender pulls in issues belonging to epics already in the structure. The issues will be placed beneath the epics in your hierarchy.



In the example above, we built our hierarchy by adding all our epics with an Agile Board Inserter(see page 125), and then we placed our issues beneath them, using the Stories Under Epics extender.

Customize Your Extender

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



You can customize:

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

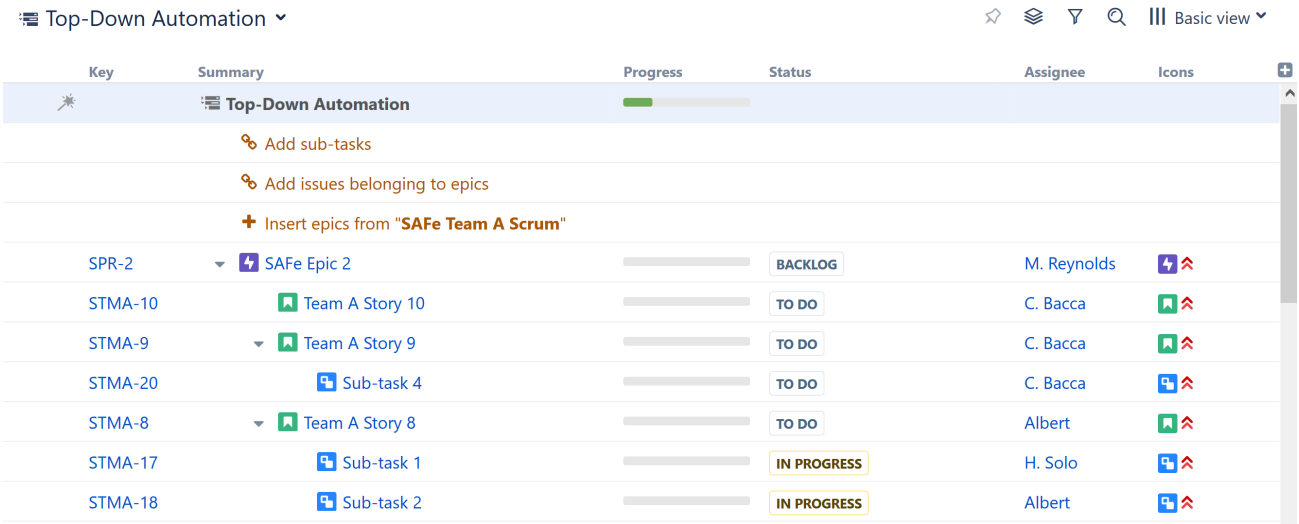
- *All levels up to 10* (default) - The extender will be applied to the first 10 levels of the hierarchy, starting from the level where the extender itself is located.
- *All levels* - The extender will be applied to the current level and all its descendants.
- *Current level only* - The extender will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

See Generator Scope(see page 169) 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.

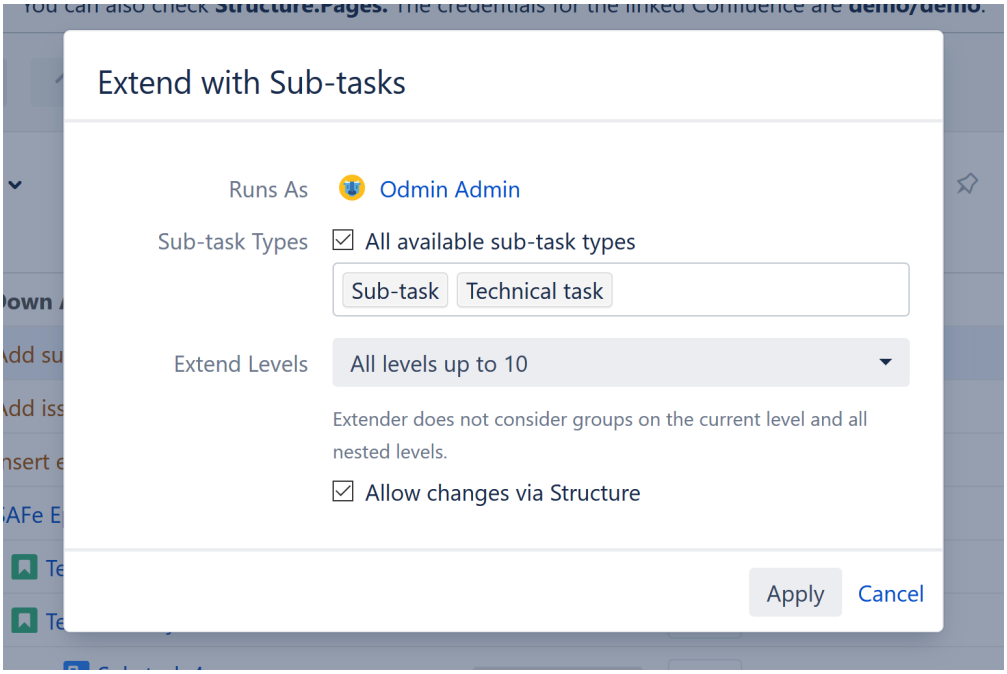


In the example above, we built our hierarchy by:

1. Adding all our epics with Agile Board Inserter(see page 125)
2. Placing our issues beneath them, using the Stories Under Epics Extender(see page 149)
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(see page 169) 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.

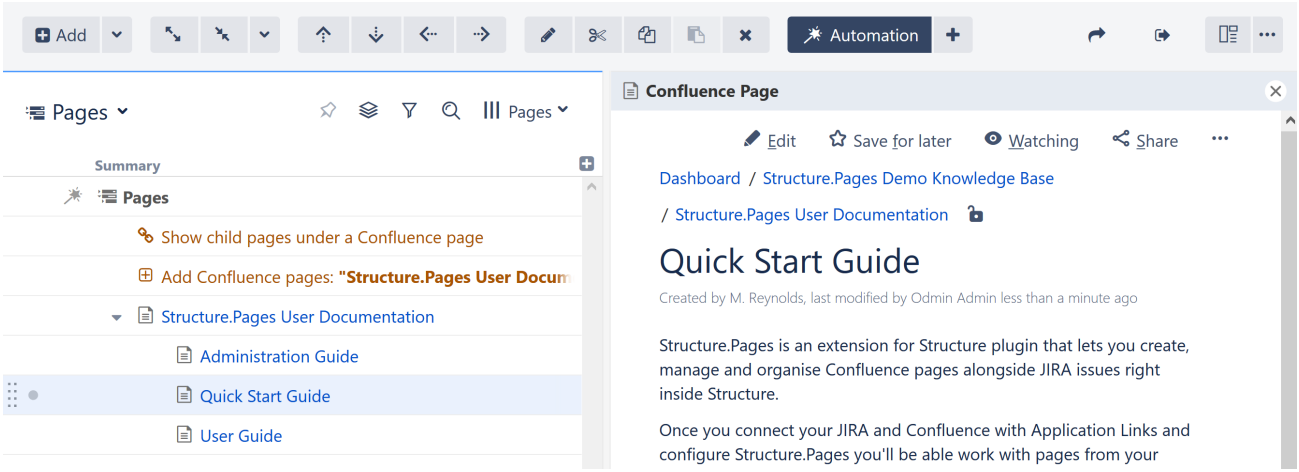
See Generator Scope(see page 169) to learn more about customizing levels.

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[20] add-on installed, you will see some additional extenders available specifically for Confluence pages.

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



Child Pages Extender

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

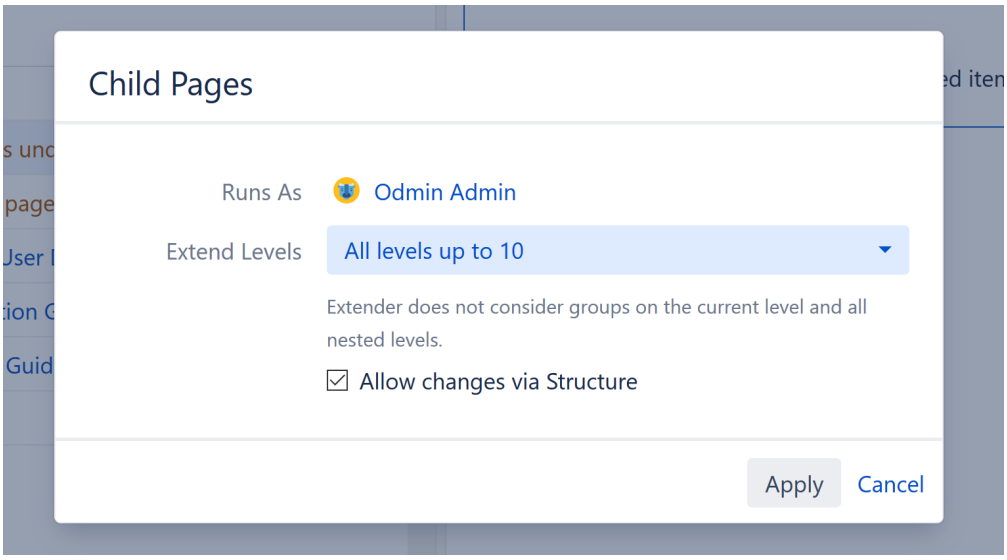
You can add pages manually, or using the Insert(see page 125) generator or Linked Pages Extender.

Linked Pages Extender

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

Customize Your Extender

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



You can customize:

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

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

See Generator Scope(see page 169) 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.

3.3.6 Effectors

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

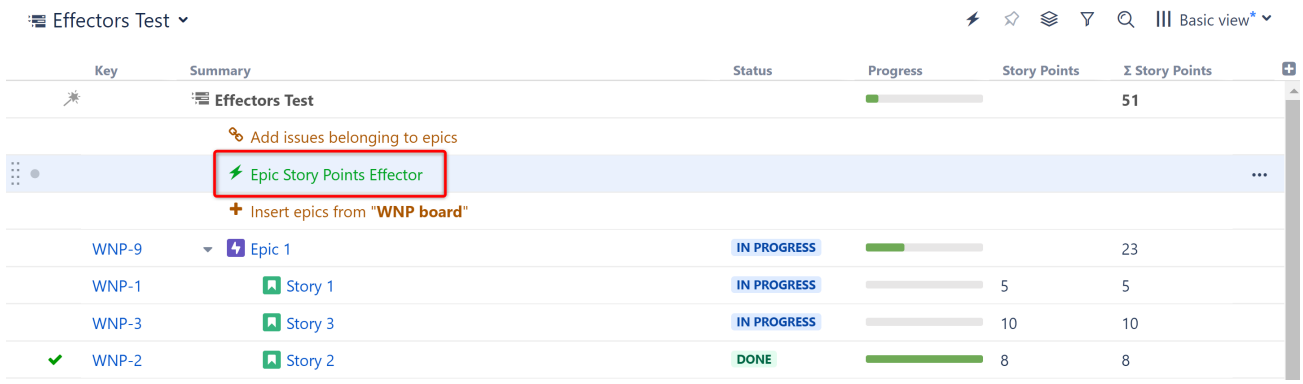
Sorry, the widget is not supported in this export.
But you can reach it using the following URL:

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

- Attribute to Issue Field Effector(see page 154)
- Status Rollup Effector(see page 157)
- Running an Effector(see page 161)
  - Running All Effectors in a Structure(see page 164)
- Revert Effector Changes(see page 165)
- Managing Effectors(see page 167)

3.3.6.1 Attribute to Issue Field Effector

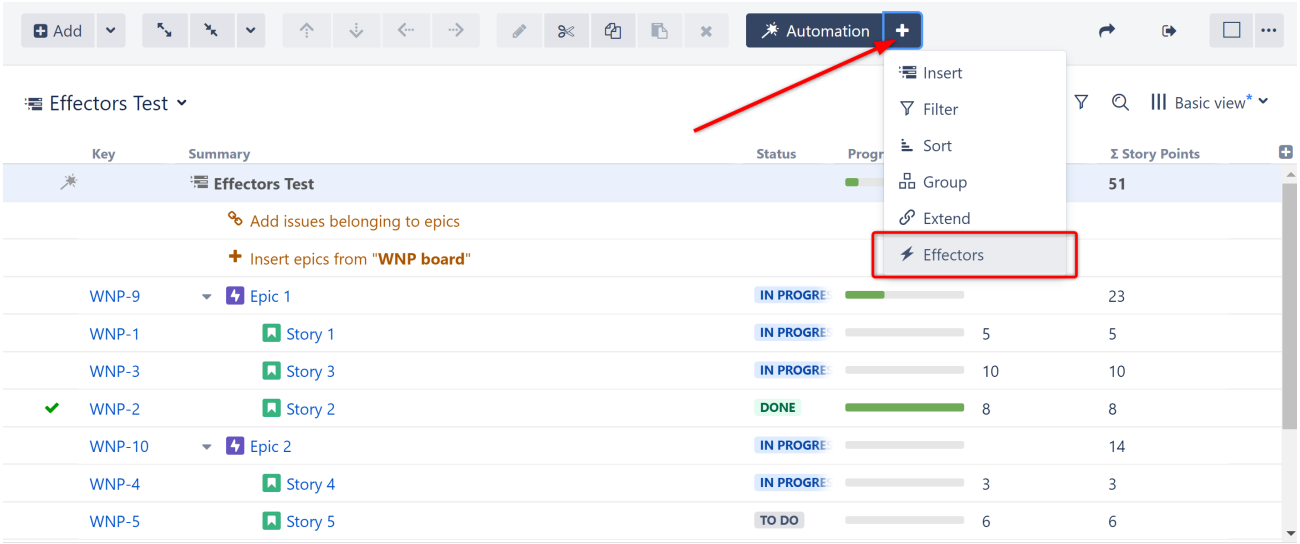
The Attribute to Issue Field Effector allows you to write the values from Structure attributes (formulas, structure-specific columns, Structure.Gantt attributes, etc.) to Jira issue fields. It can also copy values from one Jira field to another.



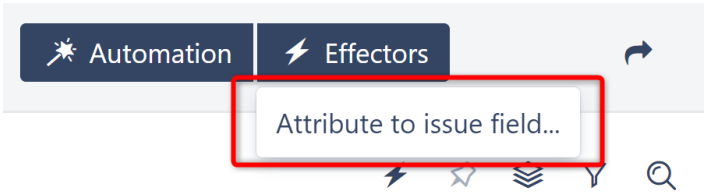
In the above example, we've added an Effector that will pull the aggregate value from the Structure-specific ∑ Story Points column and write it to each Epic's Story Points field. You can see exactly how we set it up in our Calculate Story Points for Epics and Write the Values to Jira(see page 433) article.

Adding an Attribute to Issue Field Effector to a Structure

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



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



On the Effector setting screen:

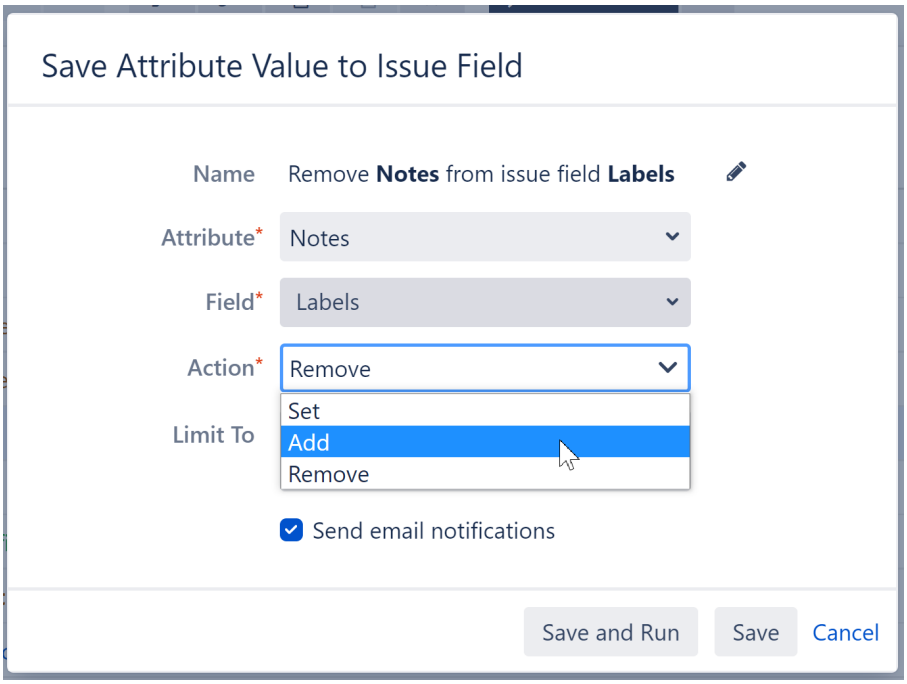
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Attribute: select the source attribute (with the values you want to write to Jira)
- Field: select the field you want the values written to
- Limit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

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

Writing to Multi-Value Fields

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

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



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

Running the Effector

Once you've added the Effector to a structure, you need to run it for changes to take place. See Running an Effector(see page 161)

Review / Revert Changes

To review or revert changes made by an Effector, see Revert Effector Changes(see page 165)

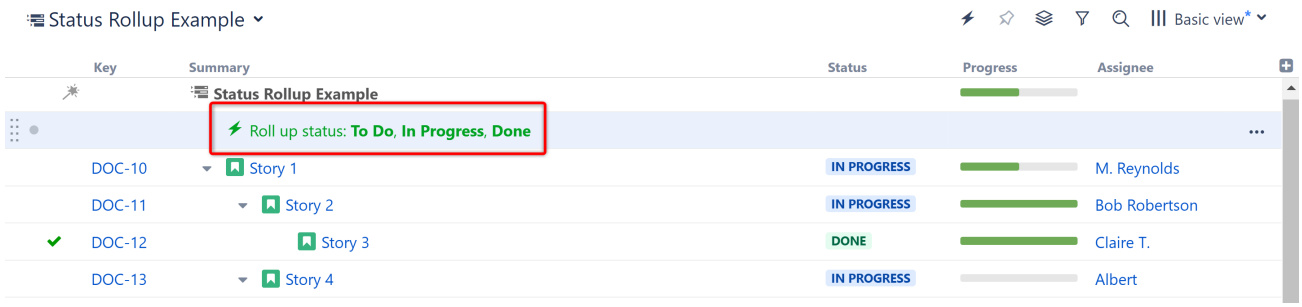
Example Use Cases for the Attribute to Issue Field Effector

Some popular uses for this Effector include:

- Add story points for epics(see page 433)
- Update issue assignees based on parent issues(see page 428)
- Calculate the cost to complete issues and projects, and write those cost to a Jira field(see page 423)
- If the Structure.Gantt extension is installed, write Gantt attributes (such as the Gantt Start Date/Gantt Finish Date) to custom Jira fields (Start Date/Finish Date).

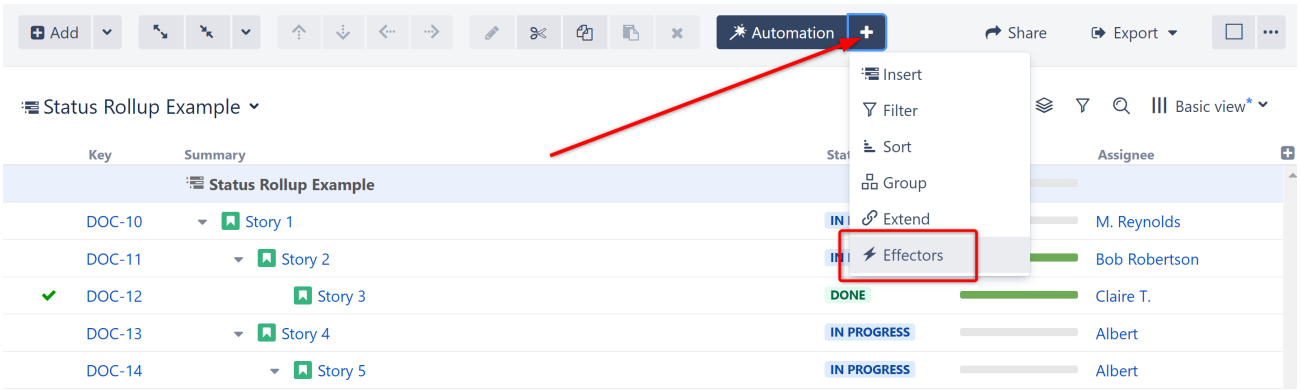
3.3.6.2 Status Rollup Effector

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

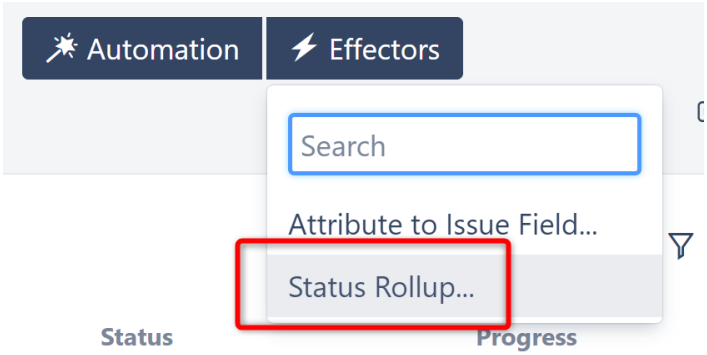


Adding a Status Rollup Effector to a Structure

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

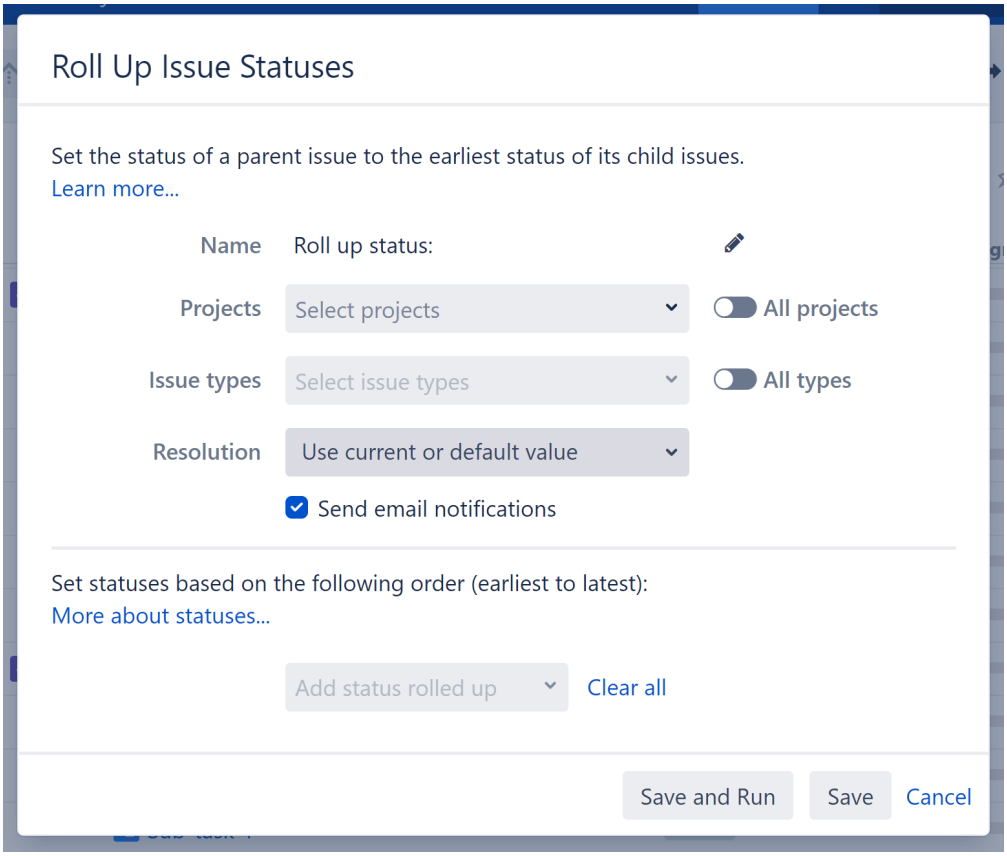


Choose **Status Rollup...**



On the Effector setting screen:

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



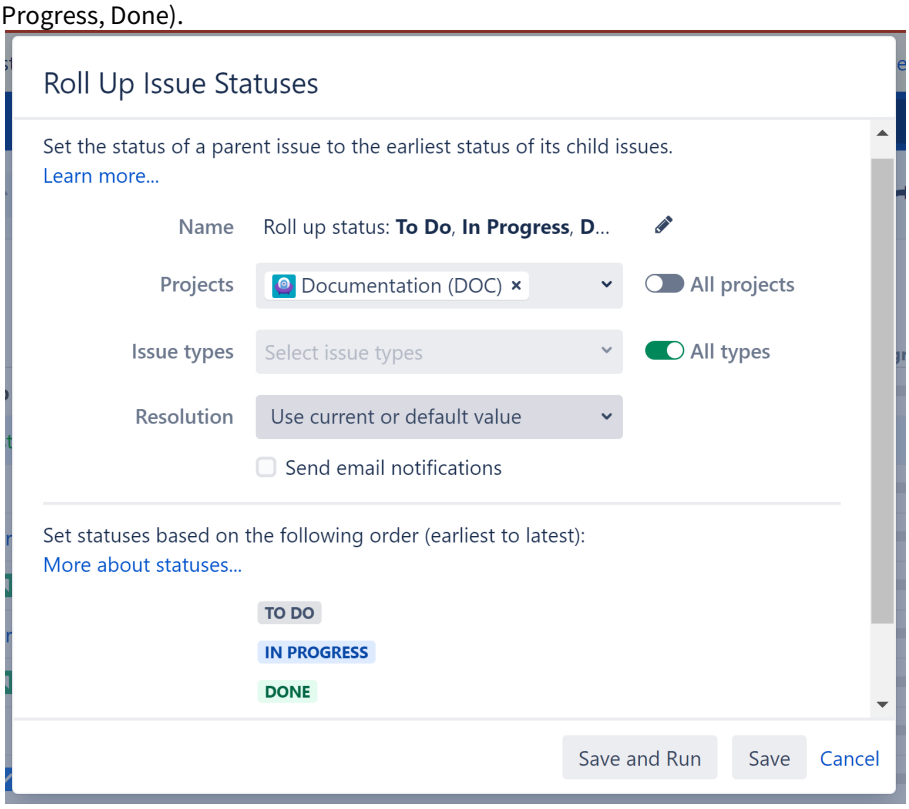
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

How the Status Rollup Effector Works

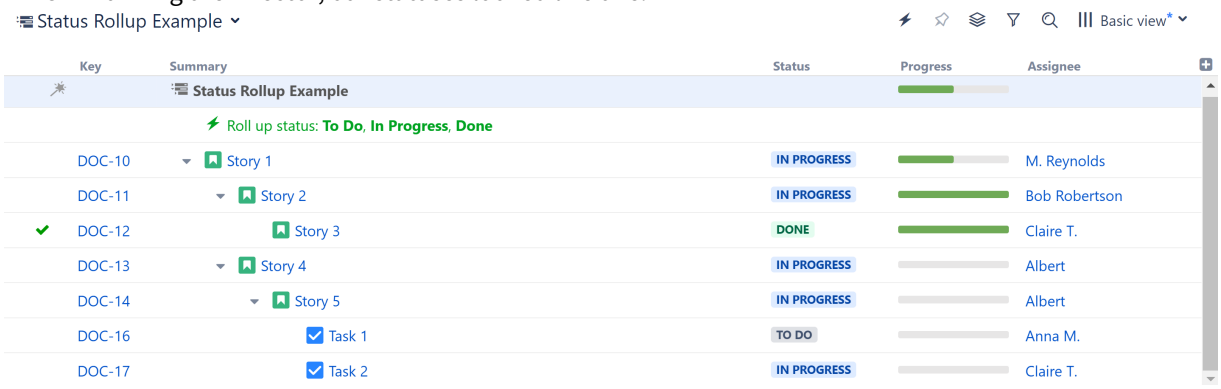
When you run the Effector, it updates the status of parent issues to the earliest status of their sub-issues, based on the project(s), issue type(s), and status order you chose in the configuration.

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

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

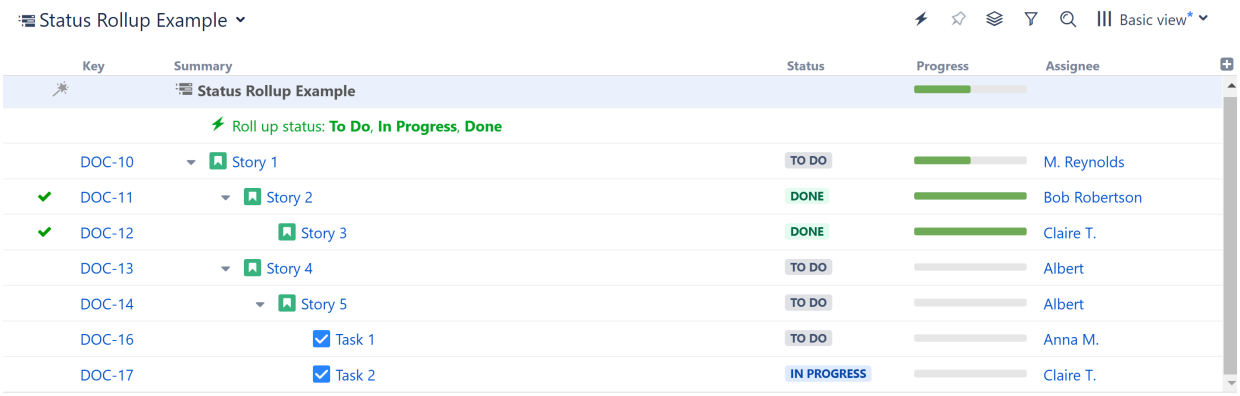


2. BEFORE running the Effector, our statuses looked like this:



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

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



Missing Status

Statuses that are not selected in the settings are not recognized by the Effector. If a sub-issue has one of the unselected statuses, it will not affect it's parent issue.

This can be useful if you want to prevent issues from being transitioned to certain statuses. For example, if you don't want the Effector to move any tasks to be moved to DONE status, simply leave DONE off your list of statuses when setting up the Effector.

Missing Issue Type

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

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

Running the Effector

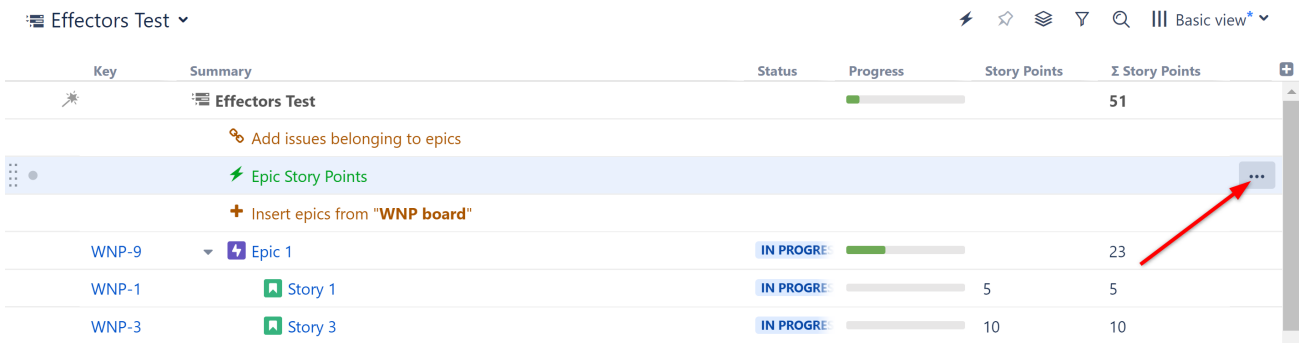
Once you've added the Effector to a structure, you need to run it for changes to take place. See Running an Effector(see page 161)

Review / Revert Changes

To review or revert changes made by an Effector, see Revert Effector Changes(see page 165)

3.3.6.3 Running an Effector

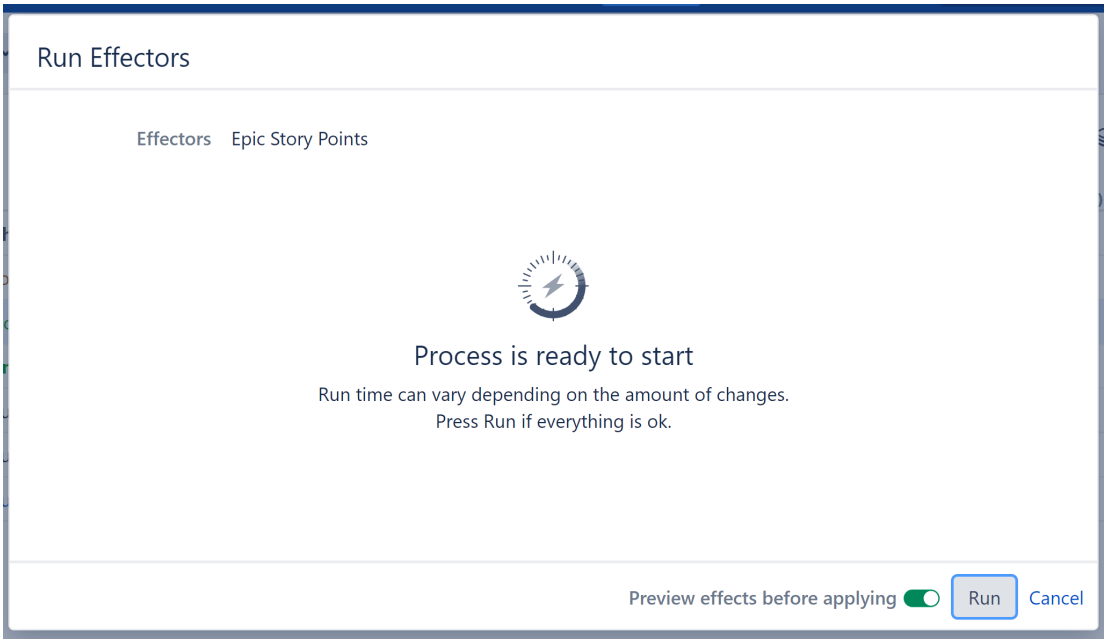
Once you create an Effector, you need to run it for any changes to take effect. To do that, locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.



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

Preview Option

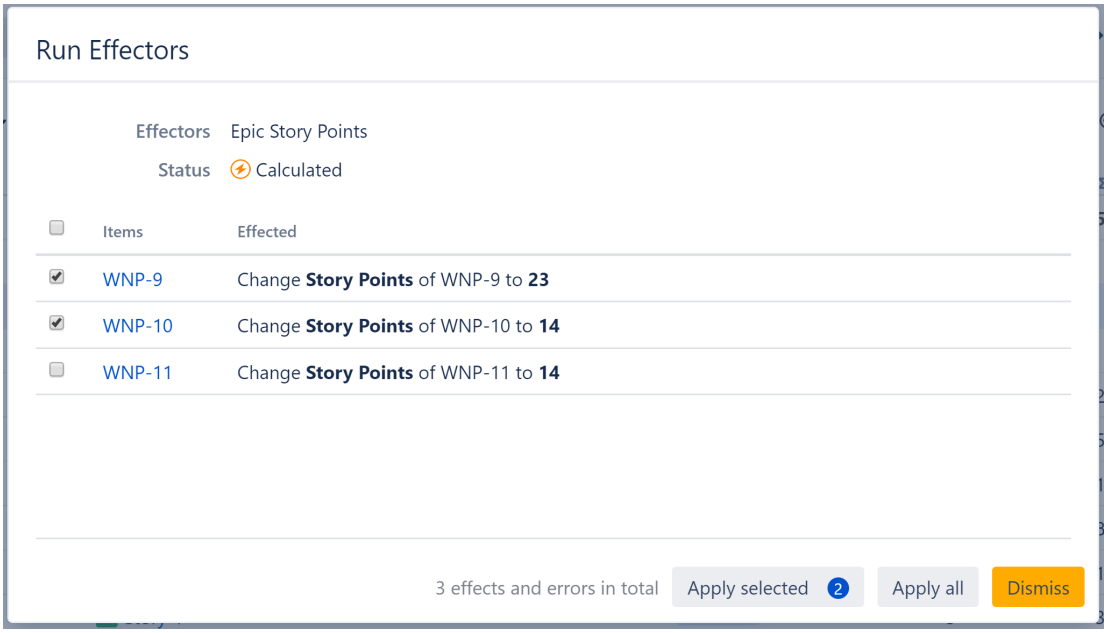
By default, Effectors initially do a Preview run, which allows you to view and approve every change that will be made by the Effector, without making any changes to Jira data. If you prefer to simply apply changes without reviewing them, switch the **Preview effects before applying** toggle off. When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



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

Preview Changes

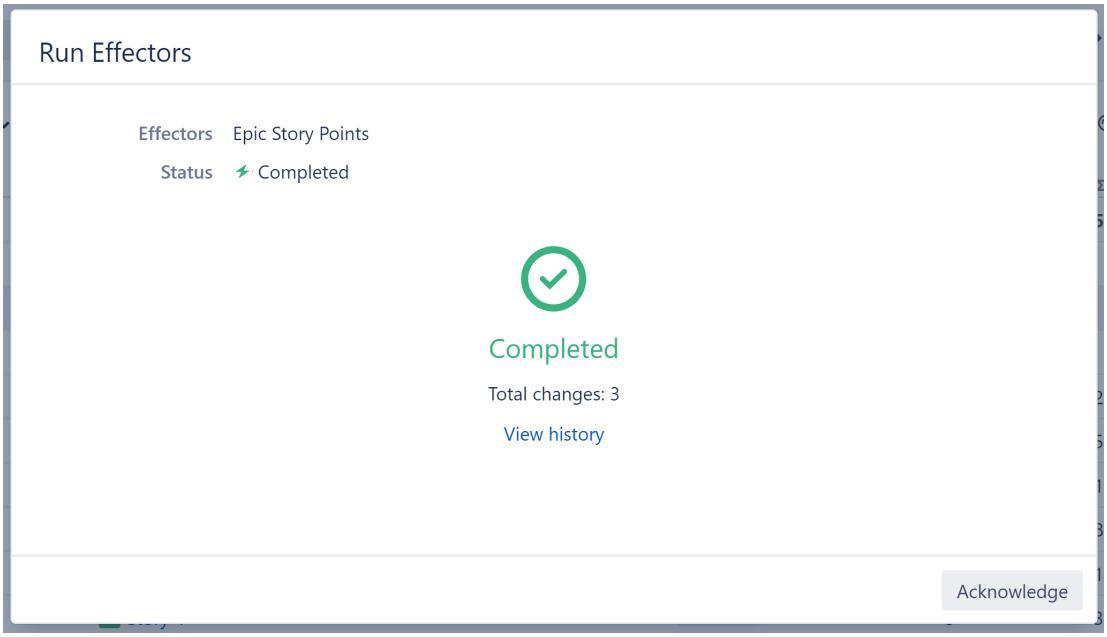
If you selected Preview effects before applying, once the preview finishes you will see a list of changes that will be made by the Effector. You can select specific changes you want made, or click **Select All** to apply all changes.



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

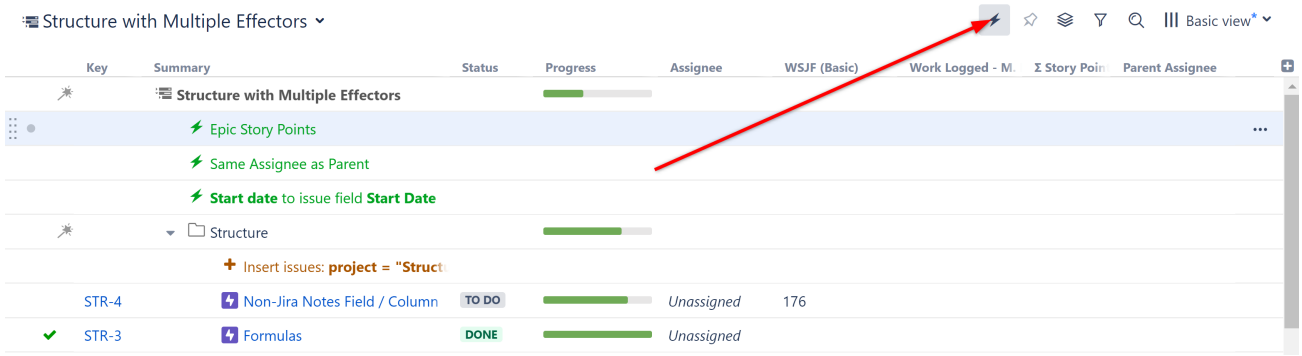
> If you click **Dismiss**, no changes will be made to Jira and the preview will be discarded. To apply changes later, you will need to re-run the Effector.

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

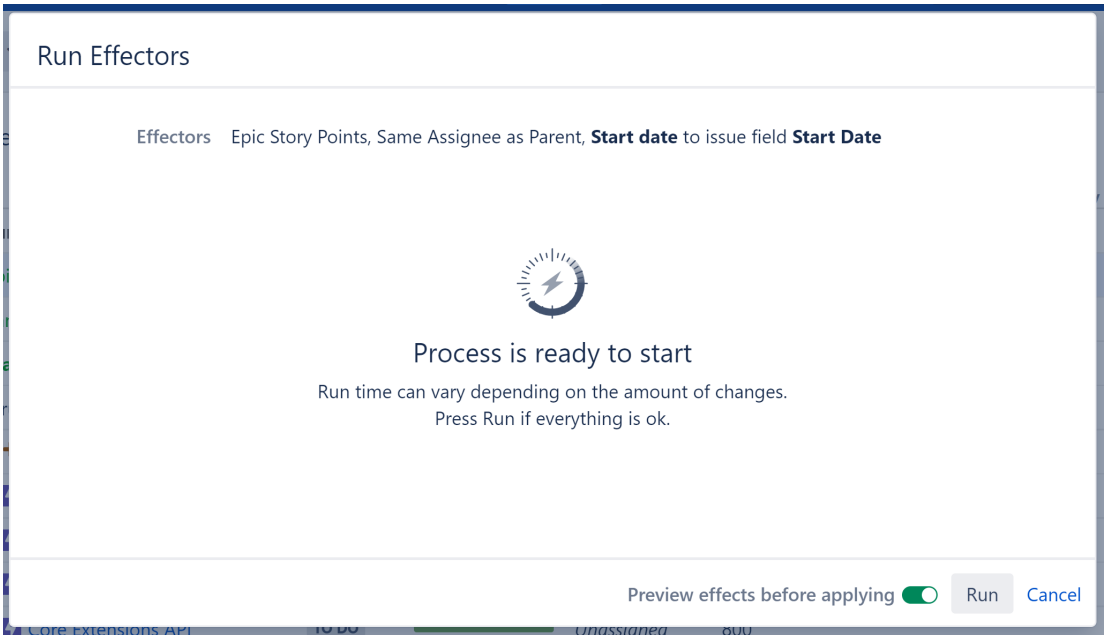


Running All Effectors in a Structure

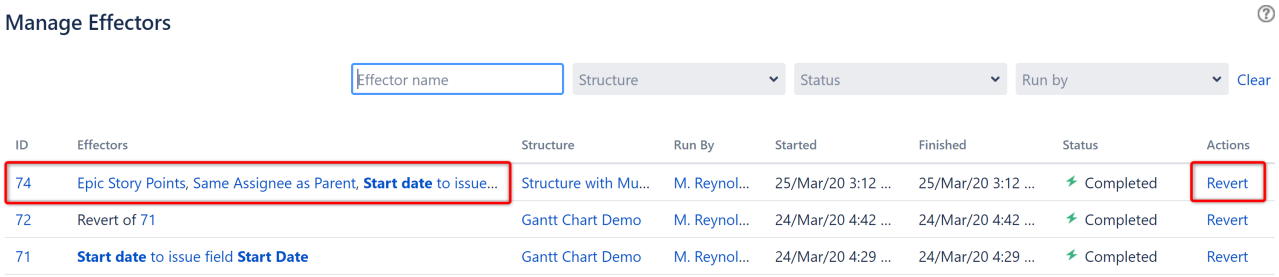
If you have multiple Effectors in a structure, you can run all of them at once by clicking the **Run all effectors in this structure** button:



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



Once the changes have been applied, you can review the operation on the Manage Effectors(see page 167) screen and, if necessary, revert any changes(see page 165).



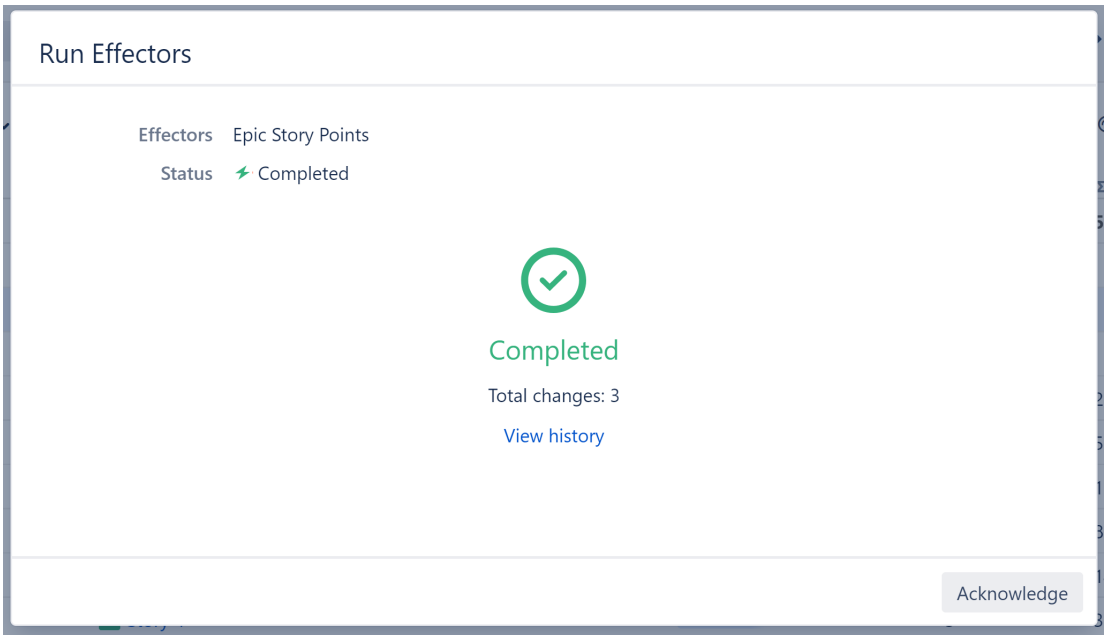
3.3.6.4 Revert Effector Changes

You can revert changes made by an Effector:

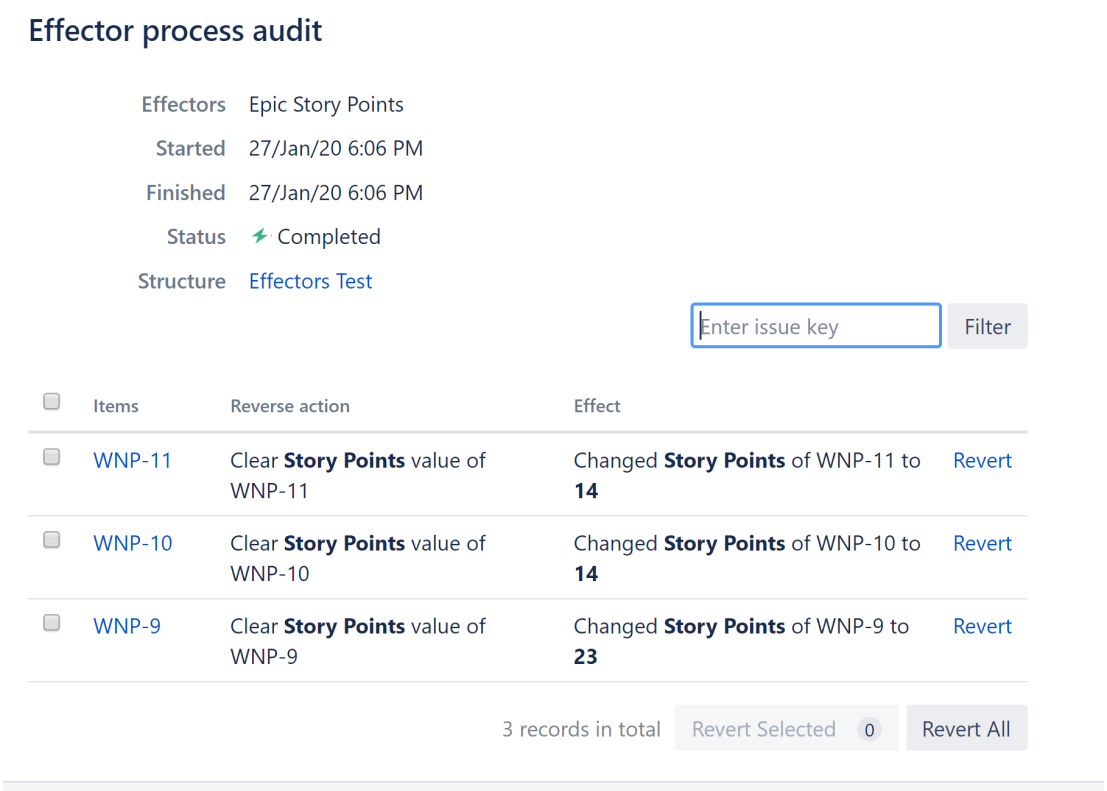
- Immedatiately after running the Effector(see page 165)
- At any time, using the Manage Effectors page(see page 166)
- You can even revert changes made during a revert process(see page 167)

Revert Effector Changes Immediately After an Effector is Run

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

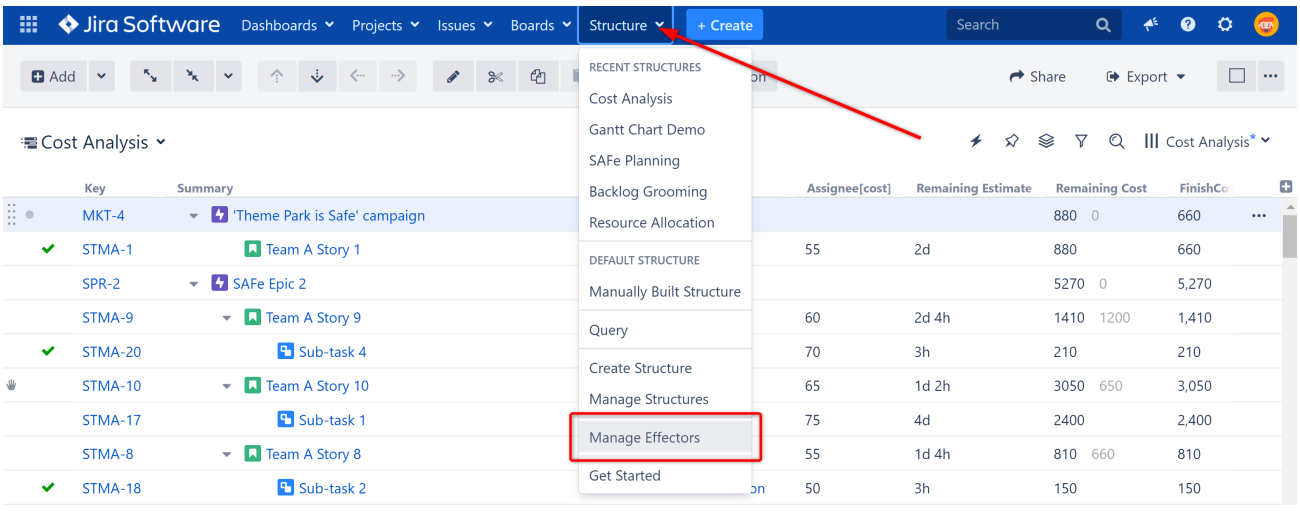


To revert any of these changes, click the **View history** link to open the Effector Process Audit screen. From there, locate the change you need to revert and click its **Revert** link. To revert multiple changes, select each and click **Revert Selected**, or click **Revert All**.

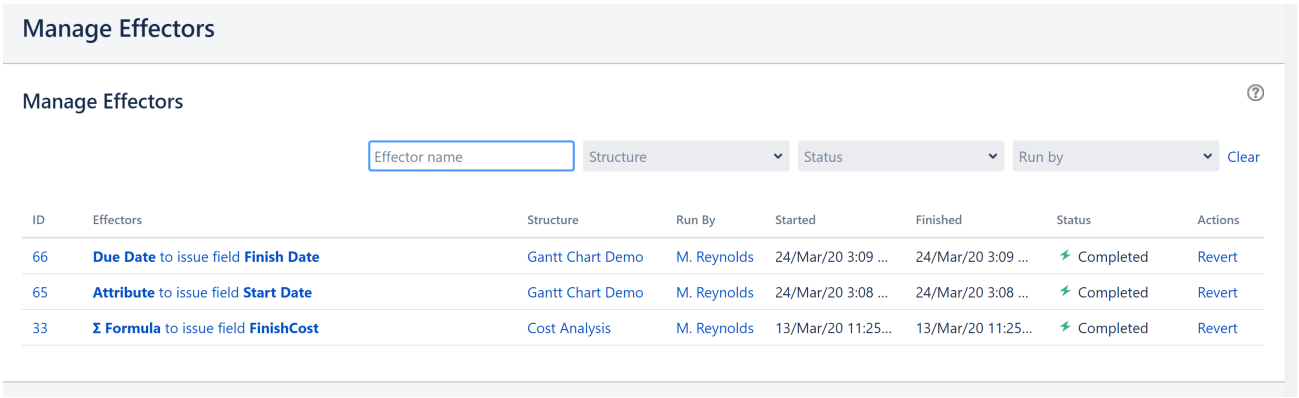


Revert Effector Changes using the Manage Effectors Page

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



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



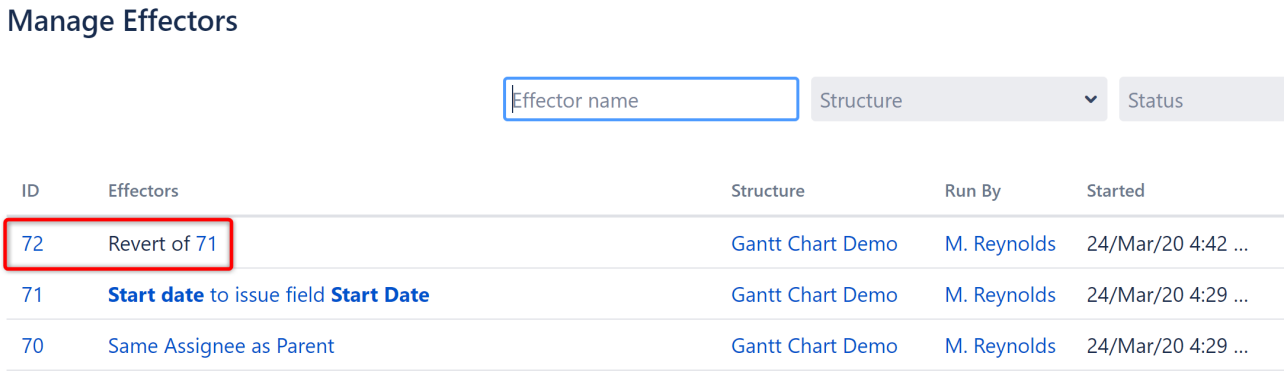
From here, you can review and revert changes from these Effectors.

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

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

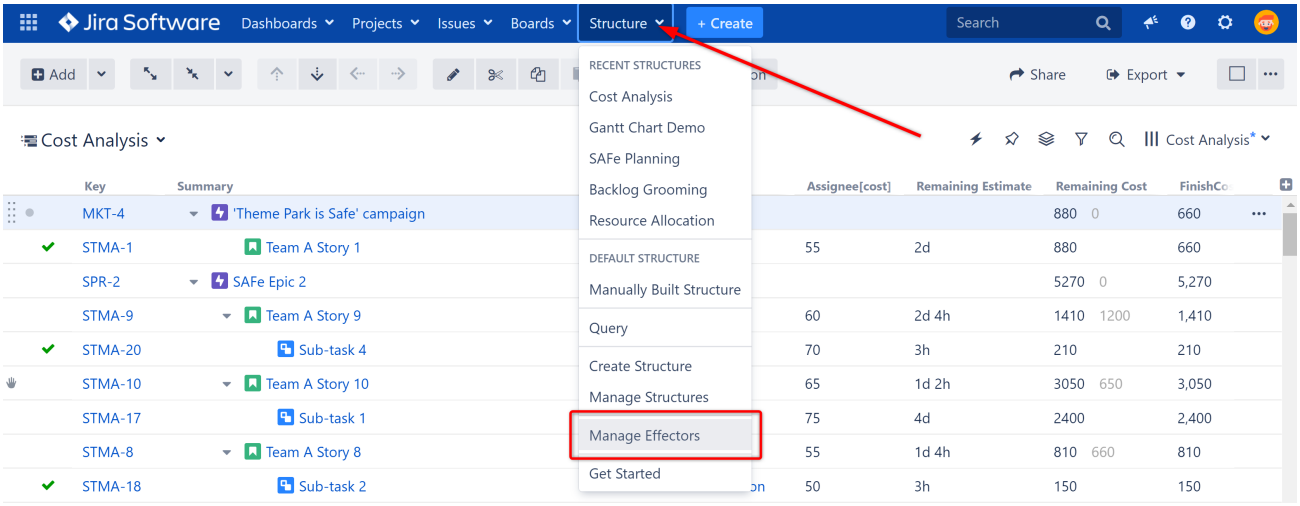
Revert Changes Made During a Revert Process

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



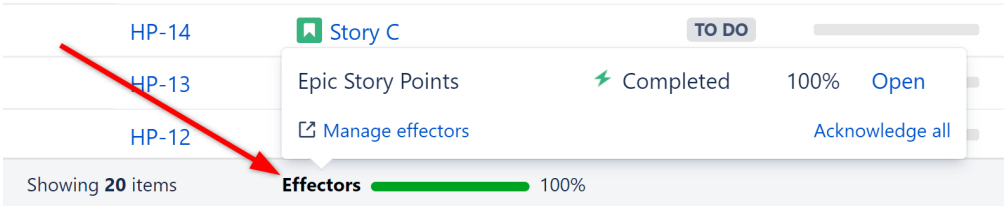
3.3.6.5 Managing Effectors

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



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

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

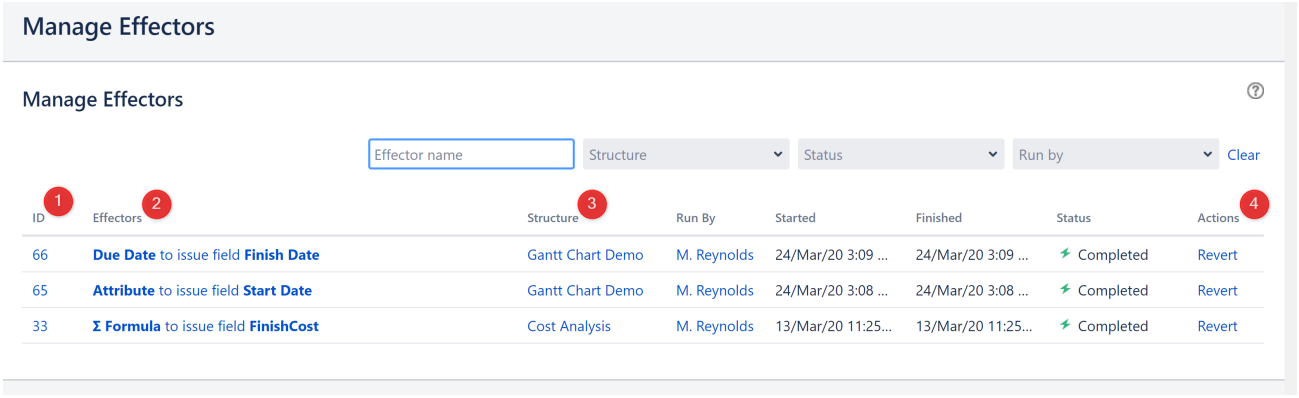


To view Effectors for a specific structure, open the Manage Structures page, locate the structure and click the number in the Effectors column.

Manage Effectors Page

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

1. Click the ID to view all changes made when an Effector was run and revert(see page 165) select changes if needed.
2. Click the Effector's name to view the Effector in its structure.
3. Click the structure name to view the structure containing the Effector.
4. Click Revert (see page 165)to undo all changes made when the Effector was run. Note: this will revert the current field values to the values prior to this Effector's run, even if other changes have been made since.



Finding an Effector Process

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

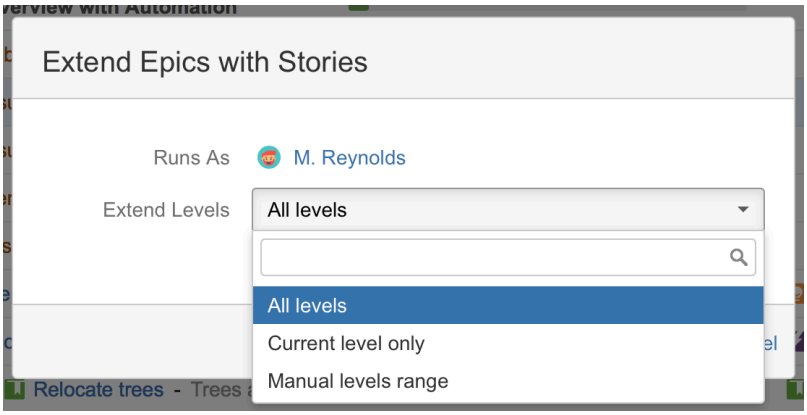
- Structure, including Structure ID
- Status of the Effector process
- Who ran the Effector process

3.3.7 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.

3.3.7.1 Manual Levels Range

The From and To fields define the range of levels to which the generator will be applied. The number entered into each field represents a level in the hierarchy, where 1 equals the level the generator is on, 2 is the next level down, etc.

Be Specific

If you wanted to pull in issues linked to the issues on level 2, set the **From** field to 2 and **To** field to 2. This will limit the generator to that specific row.

If the **To** field is set to 3, this would:

1. Pull in all issues linked to the issues in level 2, and
2. Pull in all issues linked to the new issues you just pulled in (because they will be placed on level 3).

You can also leave the From or To field blank:

- When the From field is blank, the generator is applied from the current level to the level indicated in the To field.
- When the To field is blank, the generator is applied to the level indicated in the From field and all levels below it.

> ⚠ For Group generators, the levels created by other Group generators are not taken into account when applying the specified manual levels limitation, unless the **Consider other groups** option is selected.

Building Hierarchies

The **Manual levels range** is especially useful when you want different levels in your hierarchy to have different types of relations. For example, you may want the top level and 2nd to be connected with issue links, the 2nd and 3rd with epic links, and the 4th level to be sub-tasks.

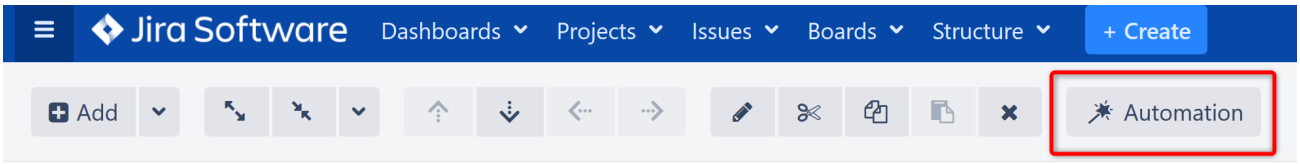
In this case, you will have three generators added under the root of the structure with the following **Levels** settings:

1. Linked Issues Extender (see page 147)working only on the top level - **Current level only**
2. Stories Under Epics Extender(see page 149) working on the second level - **Manual levels range**: from 2 to 2
3. Sub-tasks Extender(see page 150) working on the third level - **Manual levels range**: from 3 to 3

> ⓘ If you built your entire structure using Automation, you will actually have four generators - the first generator will be an Insert generator.

3.3.8 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.

3.3.8.1 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.

3.3.8.2 Change Settings

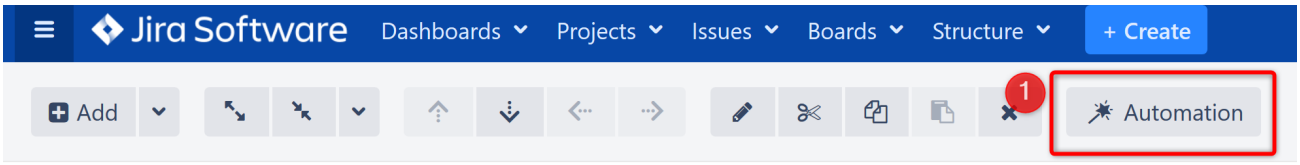
To change a generator's settings:

1. Double-click the generator or use the **Edit** button in the Structure toolbar.
2. Make the required changes and click **Apply** to save them.
3. Click the **Automation** button to hide generators (optional).

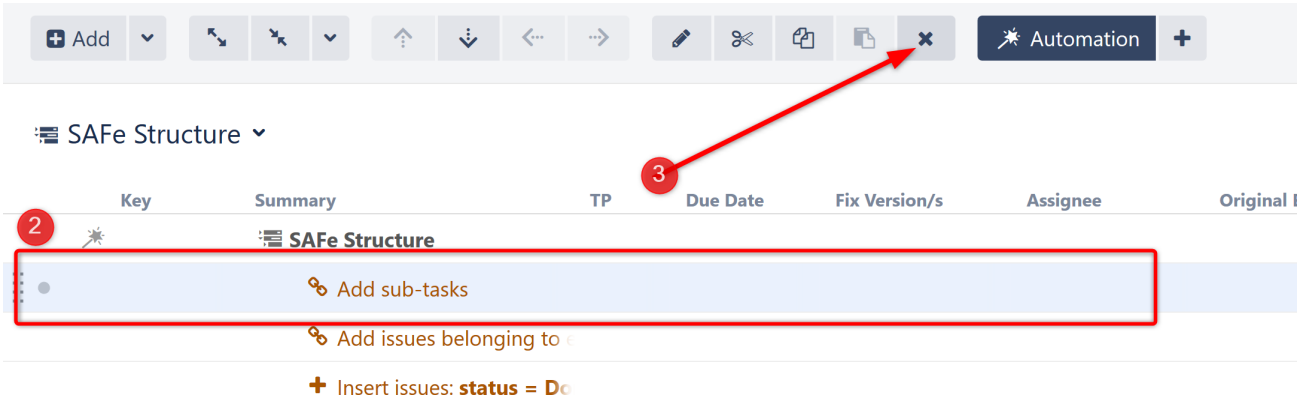
> ⓘ To learn more about the options available for each type of generator, see the appropriate article in Types of Generators(see page 124).

3.3.9 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.

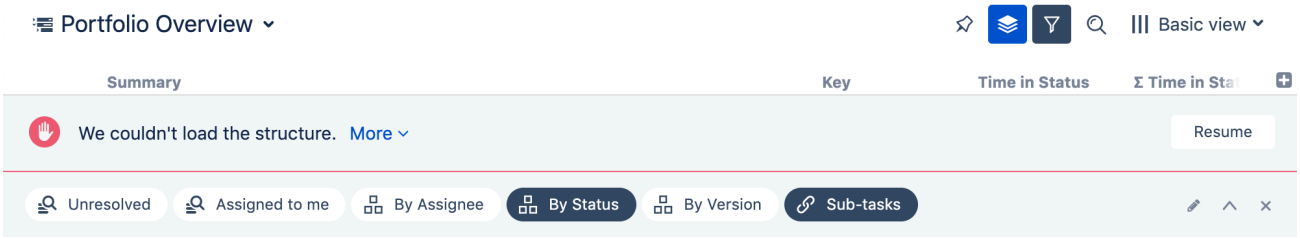


Click the **Automation** button to hide the generators (optional).

3.3.10 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.



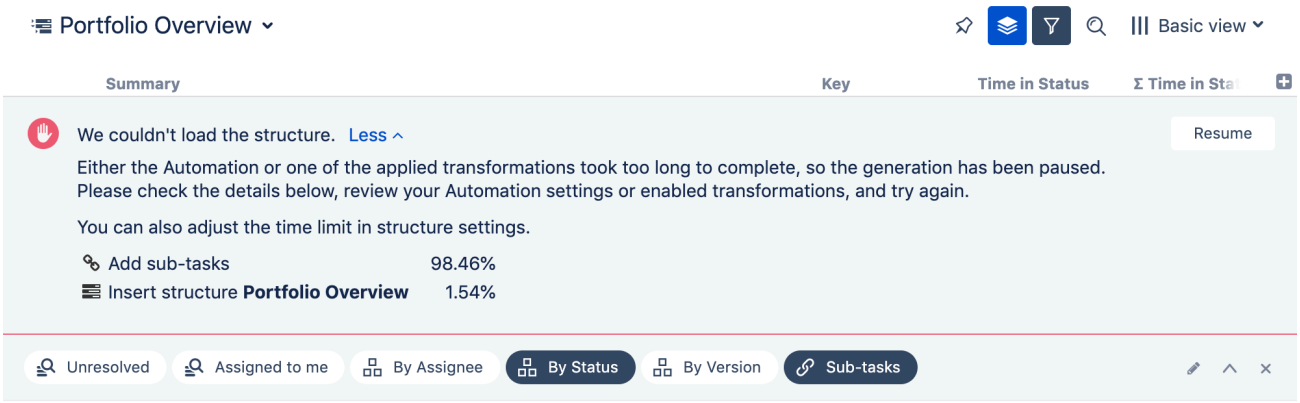
If a timed-out structure is addressed in the 'structure()' JQL function, the JQL search of that query will return an error message.

3.3.10.1 Resuming Generation

To find out which generators or transformations are working too slow, click **More** on the notification banner. Additional information will be shown, displaying the overall percentage of time that each generator took before generation was paused. The highest number will indicate the slowest generator.

If the generation process stopped while running the generators, which are part of the Automation settings, you will see the list of all generators configured and the percentage of time they consumed. This structure will also be marked as paused (see sections below for details).

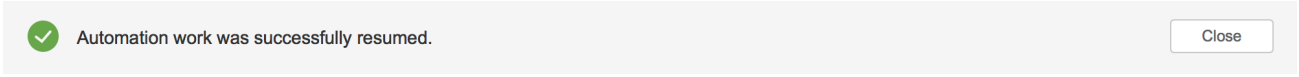
If the part of the structure created by Automation was loaded successfully and the timeout was reached while executing the transformations, you will see one Insert generator that represents the part of the structure that was added by Automation (Insert structure) and then you will see the list of all the transformations that finished successfully and the one on which the generation process was paused. This means if you had more transformations applied after this last one, they won't be mentioned in this list as they haven't been executed.



After reviewing and adjusting the automation and transformations, click **Resume** on the banner. If a structure still takes too long to generate, generation will be paused again, and the notification banner will update the 'More' section with relevant details. More actions from a user will be required, until all of the existing generators will work within the time limit.

> Deleting a generator or transformation from a structure doesn't resume generation. You must first click the 'Resume' button. This allows you to perform multiple actions before resuming.

When generation is resumed successfully, the structure will be updated with content and the banner's message will be changed accordingly. At this point, you can close the banner.

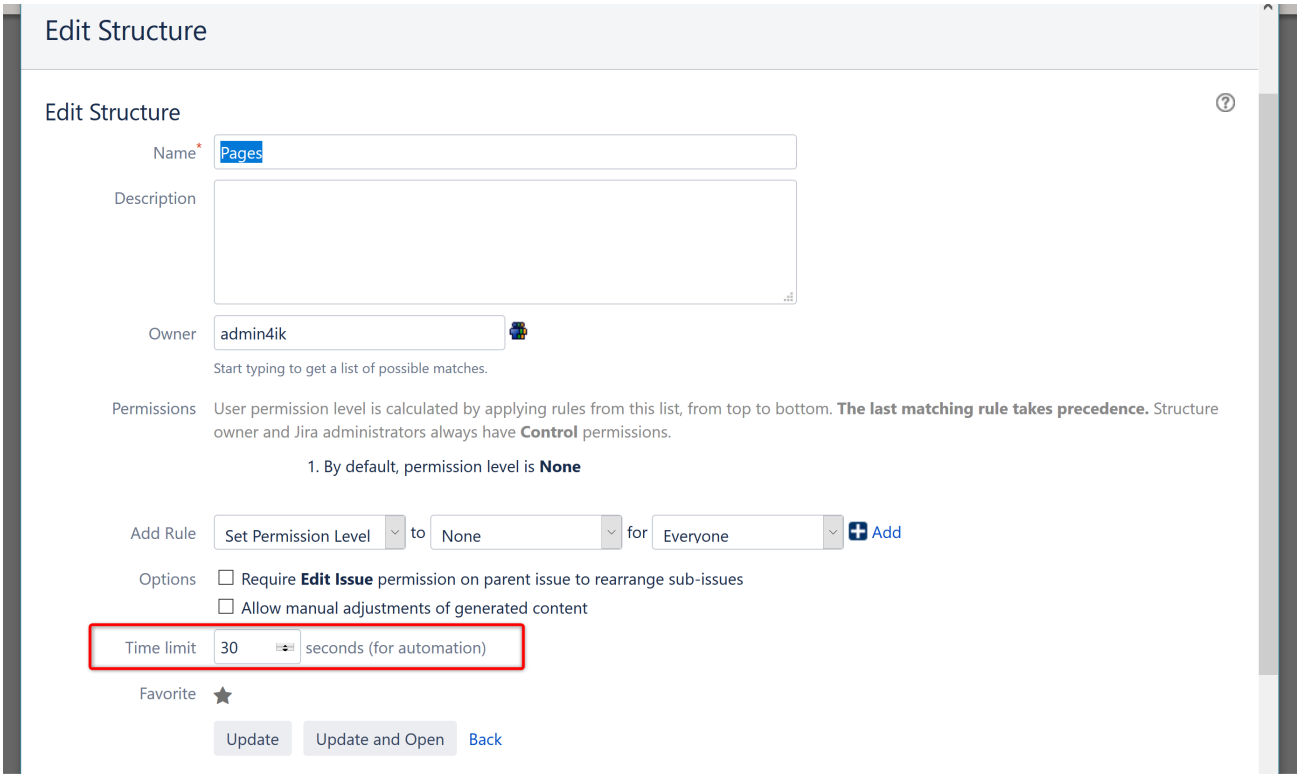


3.3.10.2 Changing the Allowed Generation Time for a Structure

If generation is paused by a generator or transformation that you consider reasonably configured, you can increase the default time limit of 30 seconds and let the structure generate for a longer time period.

To edit the time limit:

1. Go to **Structure** | **Manage Structures** in the Jira menu. Locate the structure you want to adjust (in most cases, this will be the Current Structure).
2. Click **Configure.**
3. Adjust the **Time limit** to the number of seconds you want generation to wait before timing out.
4. Click **Update** to apply the new settings.



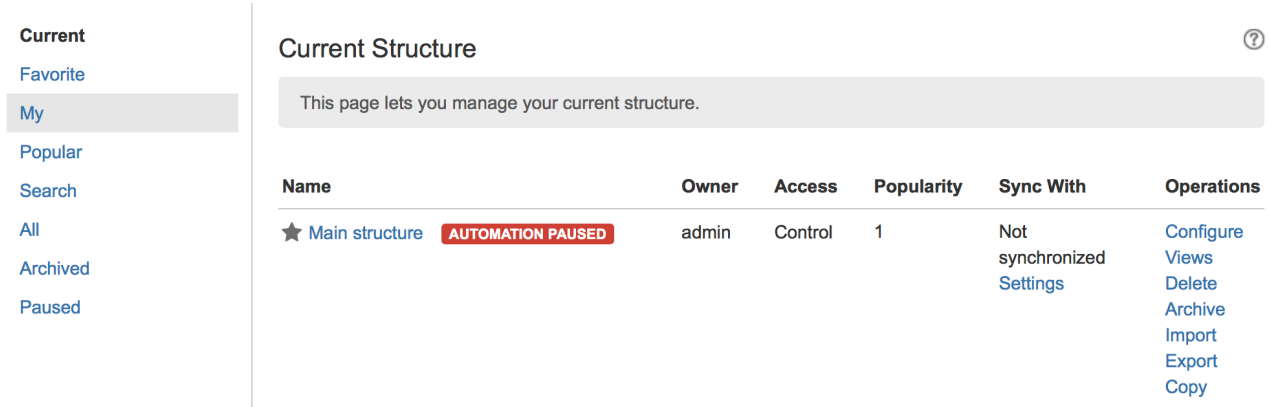
Time Limit Guidelines

When changing the **Time limit**, keep the following in mind:

- A generation time limit can't be less than 5 seconds or more than the system-wide hard limit
- Although the value is entered in seconds, the limit can be set to several minutes
- 'Control' structure permission is required to change the time limit

Identifying a Paused Structure

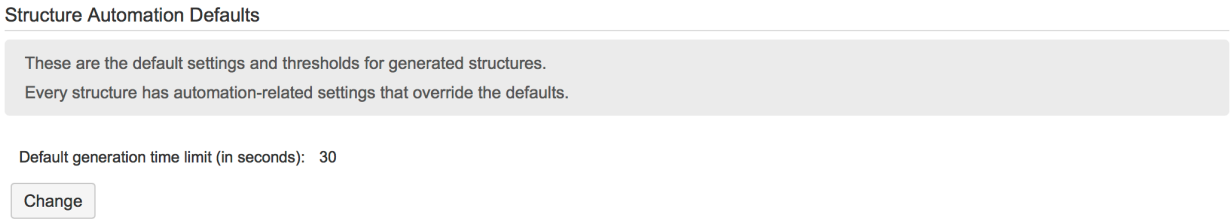
If generation is paused in a structure, the **AUTOMATION PAUSED** indicator will appear next to the structure's name on the Manage Structures screen.



You can quickly locate paused structures by clicking the **Paused** option in the left menu.

3.3.10.3 Changing the Default Generation Time Limit

To change the default generation time limit for all structures, go to **Administration** | **Structure** | **Defaults**. Look for the the **Structure Automation Defaults** section and click **Change**.



Only Jira administrators can change this setting.

If the time limit was manually set for a specific structure, it will not be changed to the default one. Only structures using the default time limit will be affected.

3.3.10.4 System-wide Hard Limit

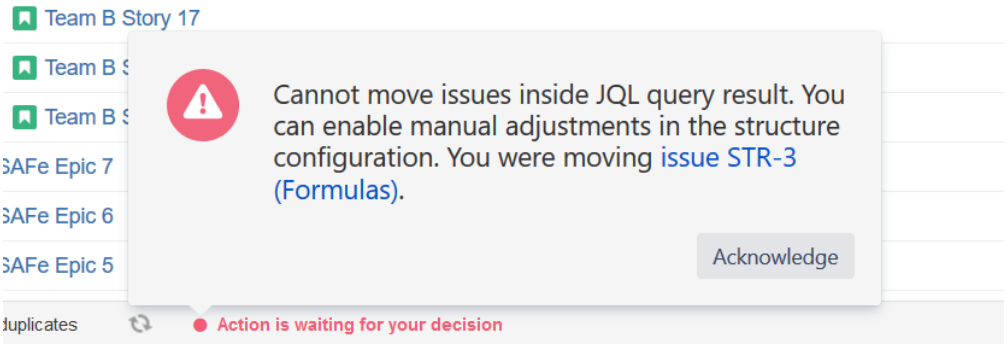
The system-wide generation time limit is initially set for 10 minutes. It can be adjusted by using Script Runner or other similar tools, or by changing the `structure.gfs.generationTimeHardLimit` property, with the Structure Dark Feature and Fine Tuning Interface(see page 499). The Generation Time Hard Limit setting accepts an integer number in seconds.

The generation time limit in a structure is only taken into account if it is less than the system-wide hard limit; otherwise it is ignored and the system-wide hard limit is used instead.

3.3.11 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(see page 122) to build all or part of a structure, the content it adds to your structure cannot be moved as freely as content which has been manually added to a structure. This is because Automation uses generators to dynamically add content from Jira, and then continuously checks that content against Jira to keep both up-to-date. If you attempt to move an item in your structure in a way that does not fit within your generators' rules, you will receive an error message.



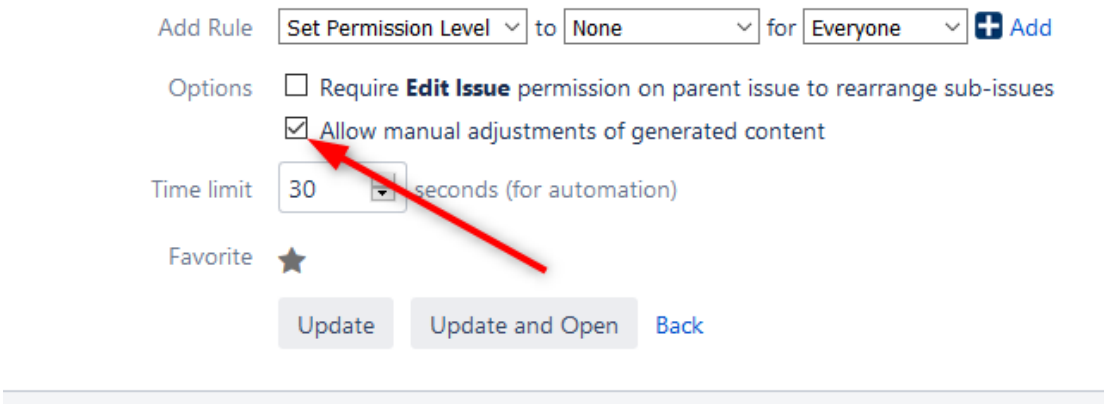
There may be times, however, when you need to move those dynamic items around freely, regardless of your generator rules. To do so, you need to enable Manual Adjustments.

3.3.11.1 Enabling Manual Adjustments

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

To enable Manual Adjustments:

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



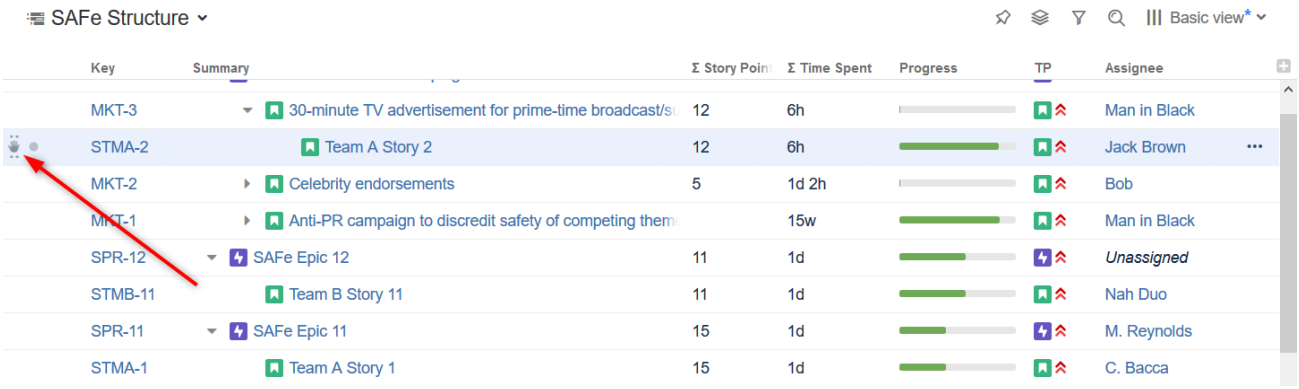
If you cannot enable/disable manual adjustments, you may not have the appropriate permissions. Speak to your Jira administrator.

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

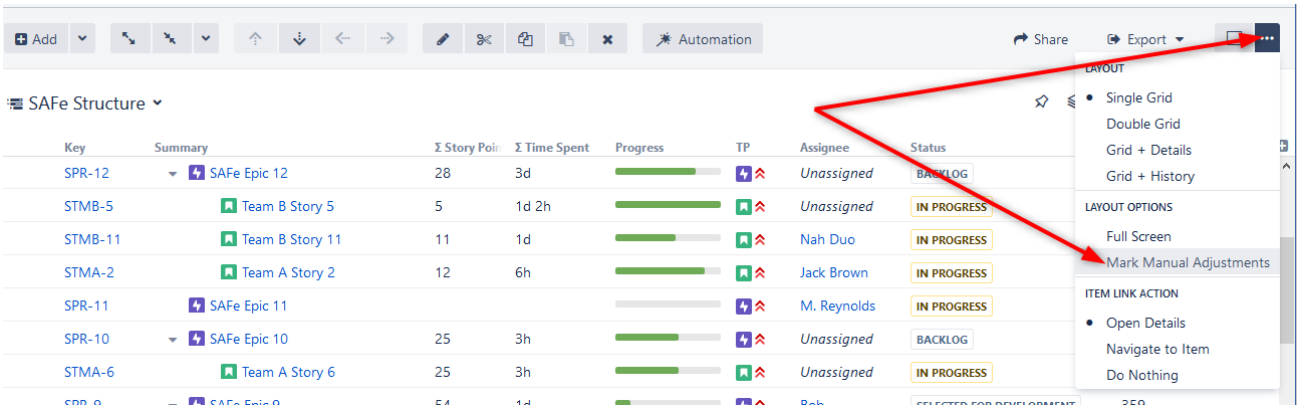
- If the move fits within the generators' rules, your content will be moved just as it was previously, and that move will be reflected in Jira.
- If the move does not fit with the generators' rules, your content will be moved using Manual Adjustments, but that move will not be reflected in Jira.

Mark Adjusted Content

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



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



3.3.11.2 Manual Adjustments are NOT Reflected in Jira

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

3.3.11.3 Special Considerations When Using Manual Adjustments

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

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

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

For more information, see Order of Operations with Manual Adjustments(see page 178)

3.3.11.4 Why is Manual Adjustment Necessary?

When you use Automation(see page 122) 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)(see page 124) used to build the structure.

This means your structure will always reflect the most recent changes to Jira, and changes you make within your structure can also update Jira. It also means some restrictions need to be in place, so content isn't moved in a way that violates the Automation rules and makes it impossible to continue syncing content with Jira. Manual Adjustment makes it possible to bypass these rules, so you can customize any structure (regardless of how it's created) to fit your needs.

To learn more about how items can be moved within generated content, see Generator Scope(see page 169).

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

3.3.11.5 Order of Operations with Manual Adjustments

The following Order of Operations is applied each time you open or refresh a structure with manual adjustments:

1. Run Generators – Structure runs your automation rules to import and organize Jira issues within your structure.
2. Manual Adjustments – Structure applies any manual adjustments over top of the generated content.
3. Transformations – Structure applies transformations after both generators and manual adjustments are made.

Manual Adjustments are applied AFTER generators. This means if anything changes within your Jira instance, it could also change (or even remove) your manual adjustment.

Here's an example:

1. You use a JQL Insert to add issues to your structure.
2. You then move Issue A under Issue X. You can't normally move issues within a JQL generator, so this requires a manual adjustment.
    - Since automation does not place issues into your structure, but rather creates rules for populating the structure (see Automation(see page 122) 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.

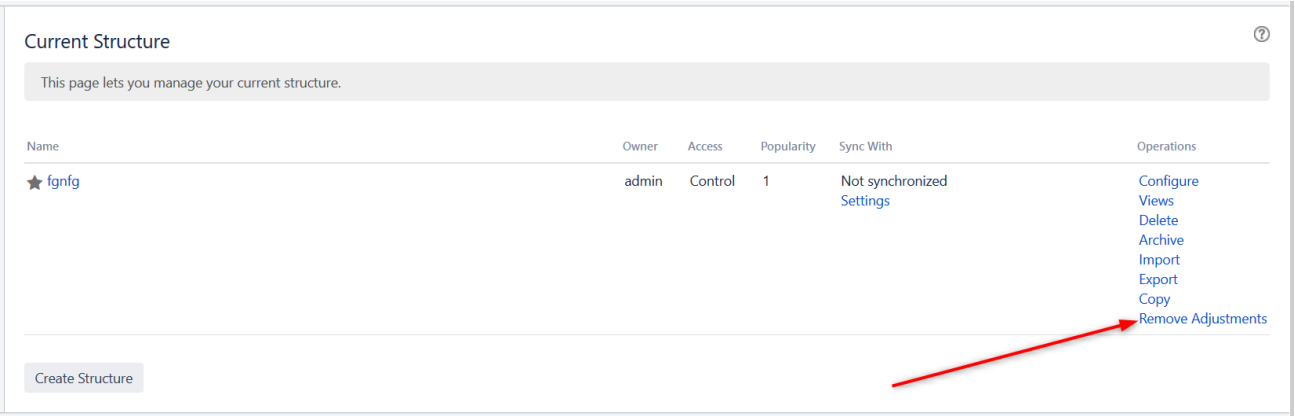
As you move it within the new set of generators:

- 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.

3.3.11.6 Undoing Manual Adjustments

To undo a single manual adjustment, simply drag the manually-adjusted item back to its original position.

To undo all manual adjustments, in the top menu go to **Structure | Manage Structure**, locate the structure you want to change, and click **Remove Adjustments**.



Removing all adjustments cannot be undone.

3.3.12 Order of Operation for Generators

3.3.12.1 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

3.3.12.2 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.

3.4 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.

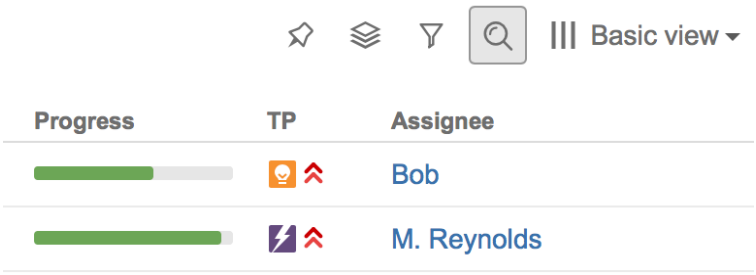
- Search(see page 180)
- Filter(see page 183)
- Transformations(see page 184)
  - Using Transformations(see page 185)
  - Quick Transformations(see page 188)
  - Default Quick Transformations(see page 192)
- Pinned Item Mode(see page 192)
- Identifying Duplicate Items(see page 194)

3.4.1 Search

The Search feature allows you to:

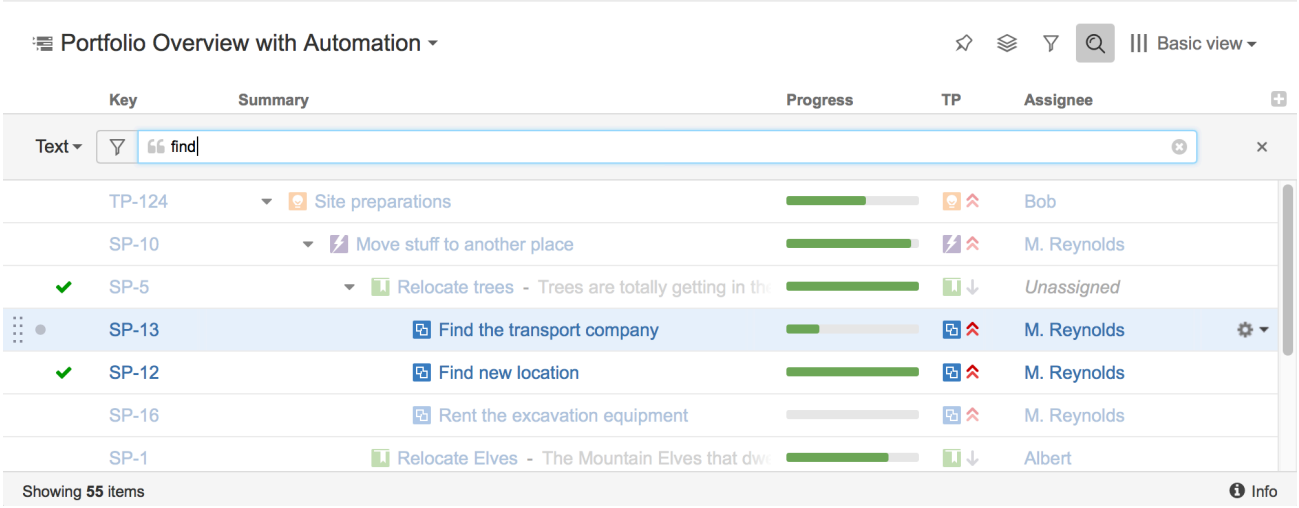
- Find and highlight issues in your structure
- Filter(see page 183) your structure so it only displays specific issues

To access Search function, click the **Search** button on the Structure Panel Toolbar.



The Search panel will appear below the toolbar. As you enter a query into the search field, results are filtered immediately, and then refined as you keep typing.

All non-matching items are grayed-out, in order to highlight your search results.



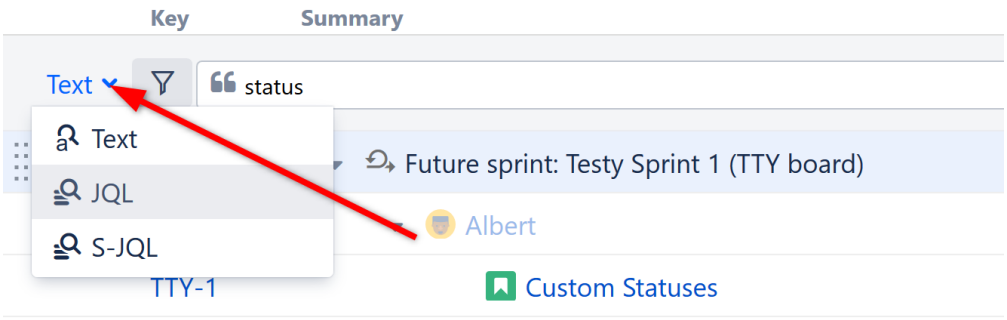
If data changes on the server, search results are automatically refreshed for the structure. So issues can be hidden and shown in the structure in real time.

Keyboard Shortcuts

- Move between matching items**: Ctrl+Alt+]** and **Ctrl+Alt+[**
- Turn on Search (or switch search mode): **Ctrl+Alt+/**
- Cancel Search & close Search panel: **Escape**

3.4.1.1 Search Modes

You can search for issues within the current structure using a Text(see page 181), JQL(see page 182), or S-JQL(see page 182) query. To switch between these modes, click the name of the currently-selected mode and select a new mode from the menu.



Text Search

Text search is selected by default. In this mode, you can specify the following search conditions:

Condition Type	Example	How it works
Simple text	*structural hierarchy*	Looks for items that have **all** mentioned words in the **Summary** field. Each word must be present in the summary or name, or the summary must have a word that **begins with** the specified word. The words may appear in any order.
Quoted excerpt	*"the quick brown fox"*	When quotes are used, the search looks for the entire phrase in the summary or name fields.
Issue keys	*MARS-1, MARS-331*	If the text looks like one or more issue keys (delimited by comma or white space), the search will return exactly these issues (if they are in the structure).

Structure relies on the Jira search engine to run text searches. The engine is based on Lucene index, which has a few peculiarities that may cause unexpected results. For example, short words may not be found. The result also depends on the Indexing Language specified in the Jira General Configuration.

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[21].

S-JQL Search

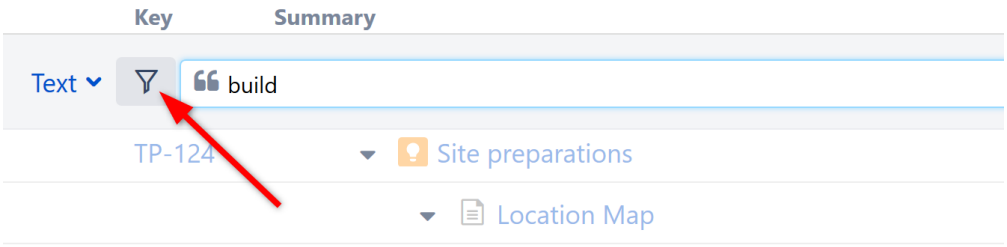
S-JQL (Structured JQL(see page 272)) is a special language that allows you to search for issues by their relationships in the current structure. For examples, `root` matches all top-level issues, `root or child of root` matches the first two levels, and `child of [priority = Critical]` matches all children of critical issues. To learn more about S-JQL conventions, see Structured JQL(see page 266).

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

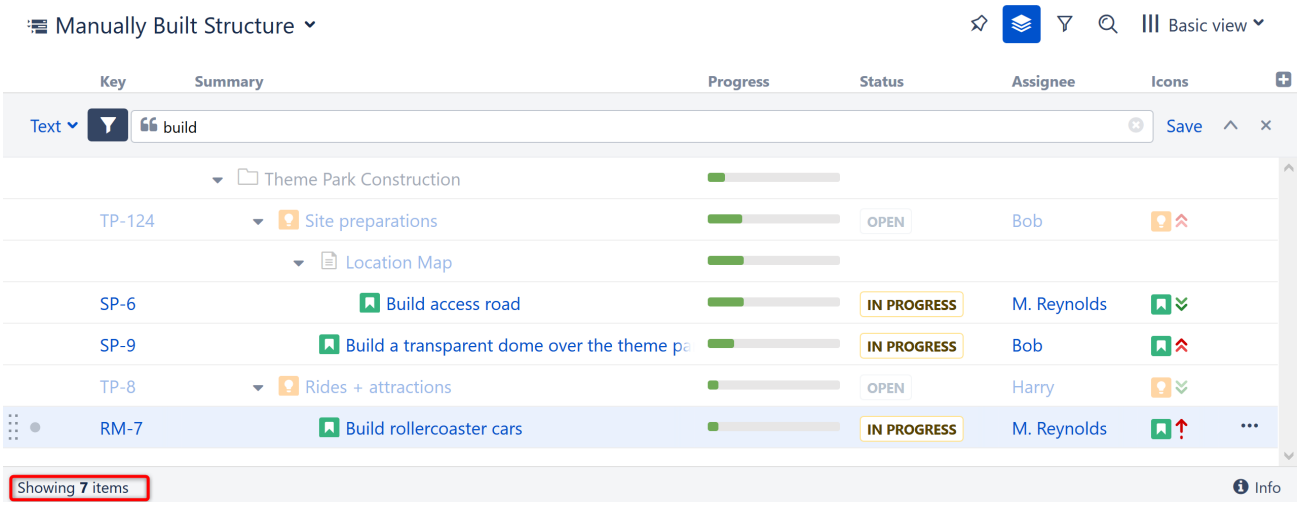
21 http://confluence.atlassian.com/display/JIRA/Advanced+Searching

3.4.2 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.

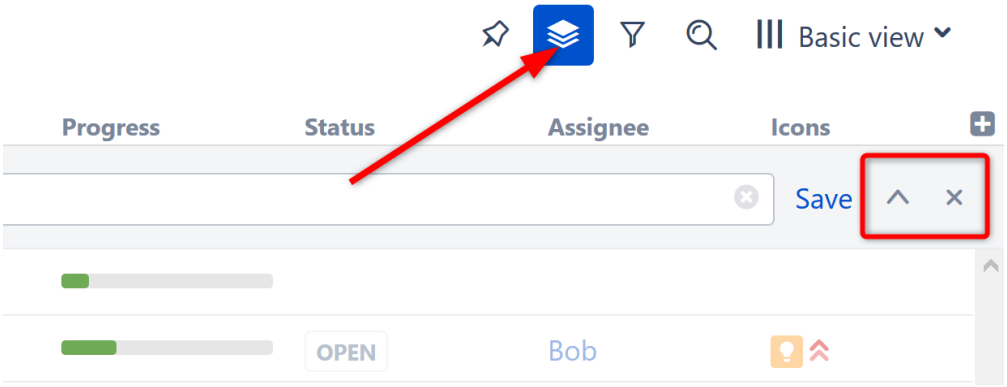


In the status bar at the bottom, you will see an updated items count.

> You can apply the filter to any text, JQL or S-JQL search(see page 180).

3.4.2.1 Filter Is a Transformation

When you apply the filter, you are also creating a Transformation(see page 185). 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.

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

3.4.3 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(see page 183) out all but those issues assigned to the current user or to sort issues by progress (or both!).

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

https://www.youtube.com/watch?v=IW0JTizto58

3.4.3.1 Transformations vs. Generators

Transformations use the same types of functions as generators(see page 124). 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.

3.4.3.2 Available Transformations

The following types of transformations are available in Structure:

- Filter
- Sort
- Group
- Extend

For more details on how each of these works, please see the documentation on Types of Generators(see page 124).

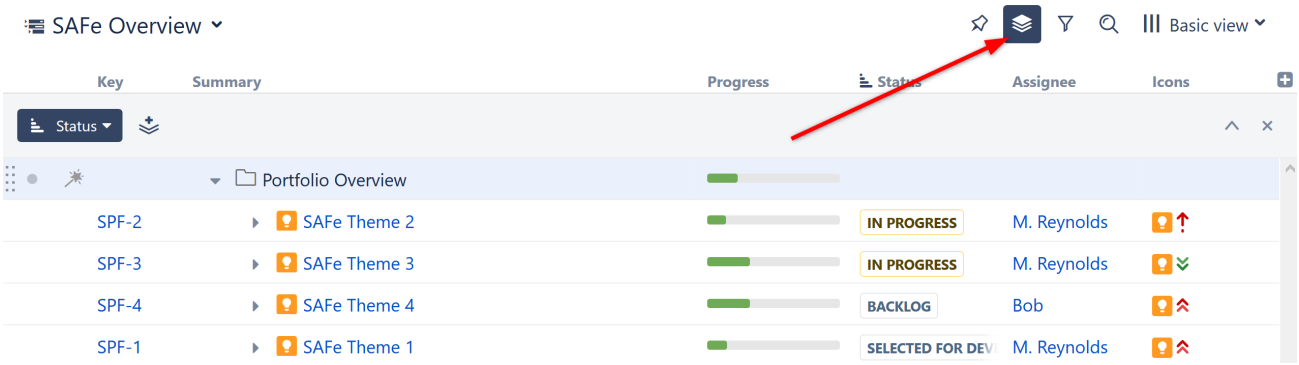
> ⚠ The Filter by Sprint transformation only affects folders, not issues, and it is applied to the whole structure. The Filter by Sprint generator is only applied to embedded sub-structures.

3.4.3.3 Working with Transformations

- Using Transformations(see page 185)
- Quick Transformations(see page 188)
- Default Quick Transformations(see page 192)

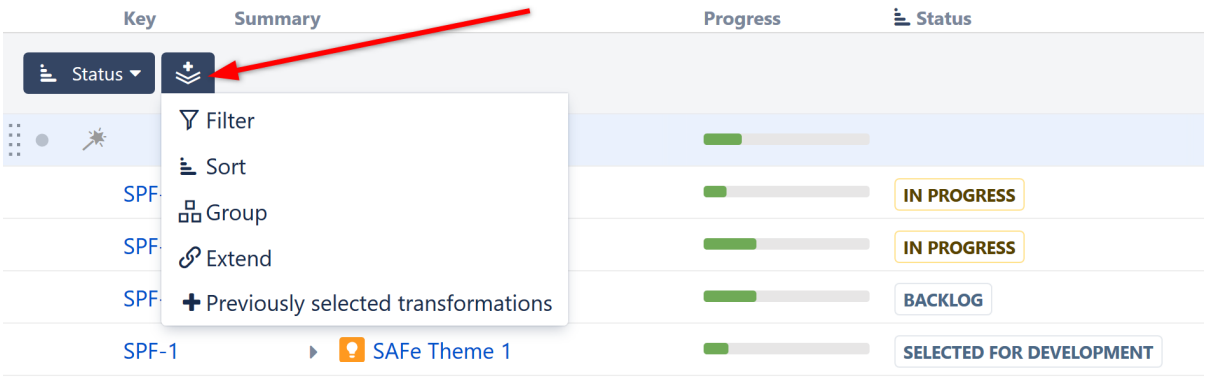
3.4.3.4 Using Transformations

All transformations can be added and modified on the Transformations Panel. To access it, click the **Transformations** button in the panel toolbar. If any transformations are currently applied, they will appear in the panel.



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(see page 124).

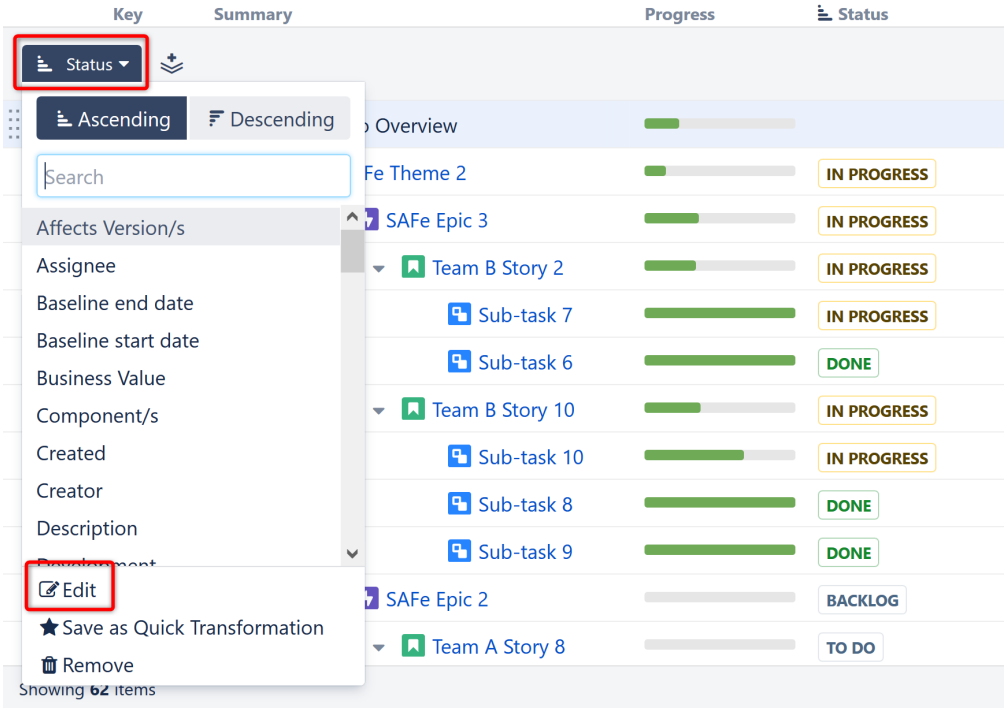
> If you add transformations and then switch structures using the Structure selector, you can apply the same transformations to the new structure by selecting **+Previously selected transformations**.

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.

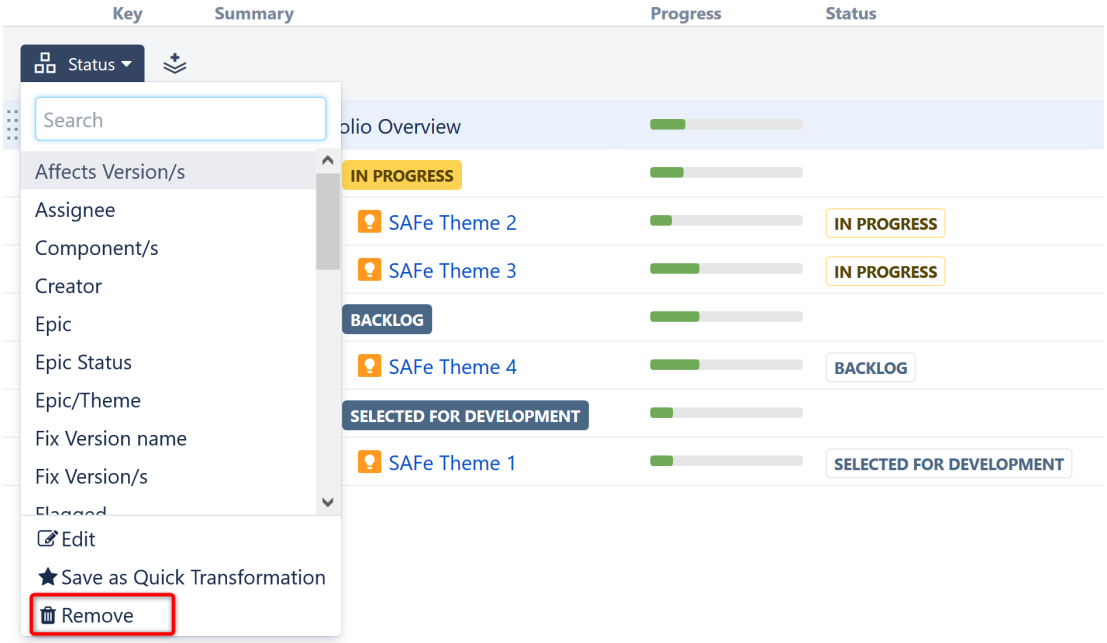


For more information about the advanced options available for each transformation, see the documentation for the corresponding generator(see page 124).

You can quickly change a Sort transformation by clicking another column header – the structure will be sorted by the new column, in ascending order. See Sorting and Filtering(see page 188) 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

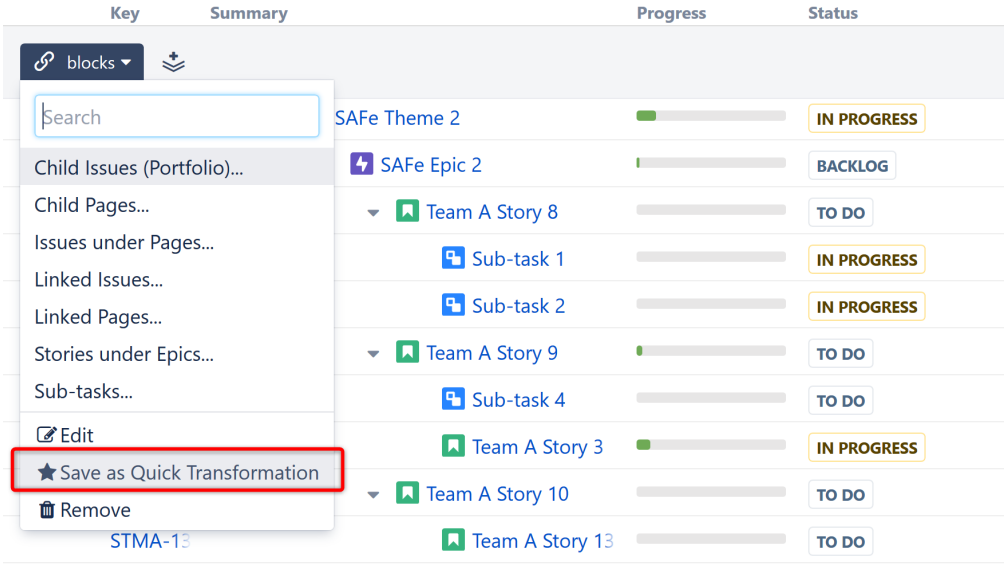
After you've configured your transformations, you can hide the panel without removing the transformations. Click the **up arrow button** on the right side of the Transformations Panel to hide it.

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(see page 188). Click the transformation you want to save, and select **Save as Quick Transformation** at the bottom of the drop-down menu.



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

Sorting and Filtering

Sort and Filter transformations can be applied very quickly, without opening the Transformations Panel.

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

- To sort in descending order, click the column header again.
- To change the scope, choose the Edit option(see page 186).
- To remove the sorting, click the **Summary** column header.

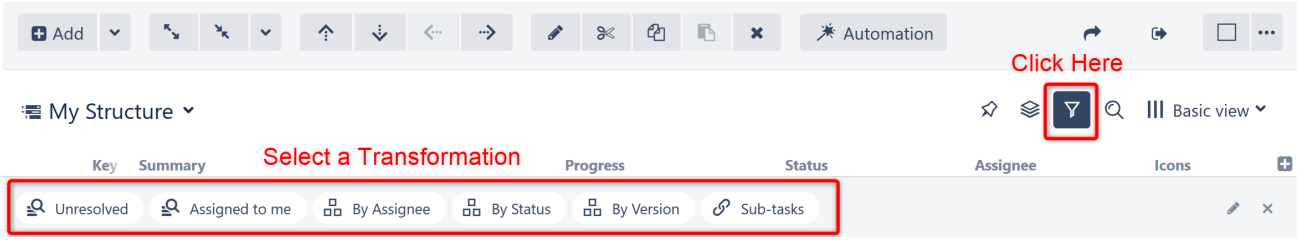
To apply a **filter**, you can run a search(see page 180) and then filter(see page 183) out non-matching items.

3.4.3.5 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.

> The Quick Transformation panel shows transformations that are associated with the current structure. See Defining Quick Transformations(see page 190) to learn how to add your own custom transformations.

Default Transformations

If the list of quick transformations was not customized for the displayed structure, the default quick transformations will be shown. Default transformations are also shown in the Secondary panel when displaying query results or other non-structure content.

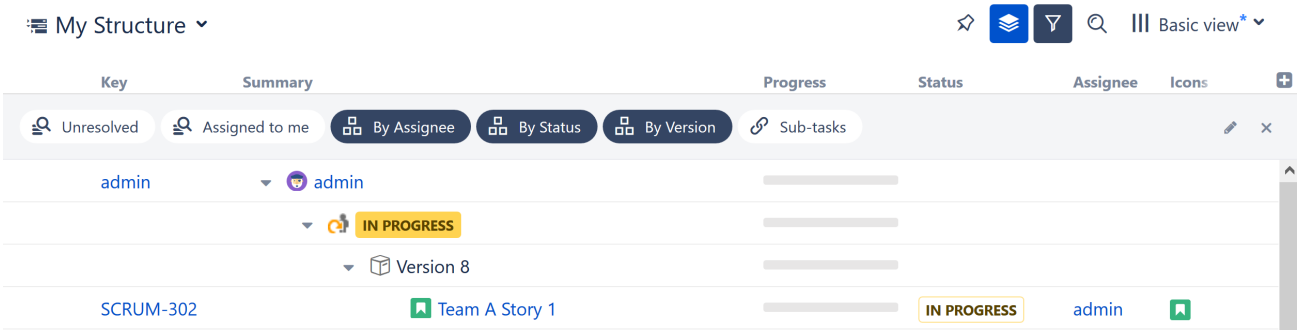
To learn more, see Default Quick Transformations(see page 192).

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:

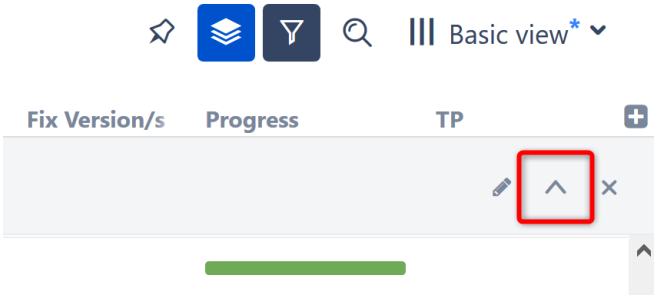
1. Group the first level by Assignee
2. Group the second level by Status
3. Group the third level by Version



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.

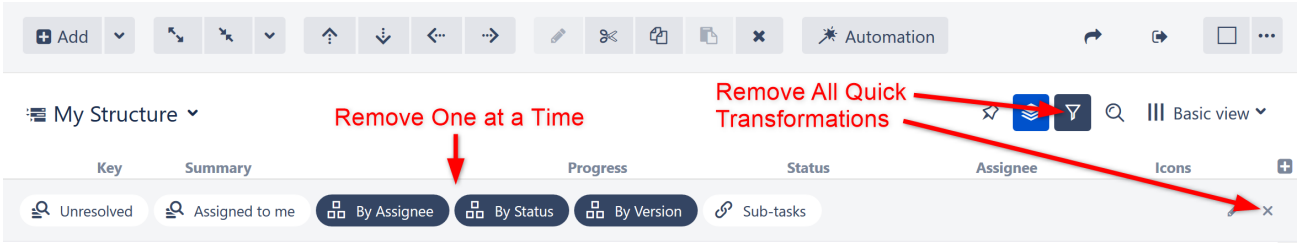


When the panel is hidden but quick transformations are applied, the Quick Transformations button will become blue .

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.



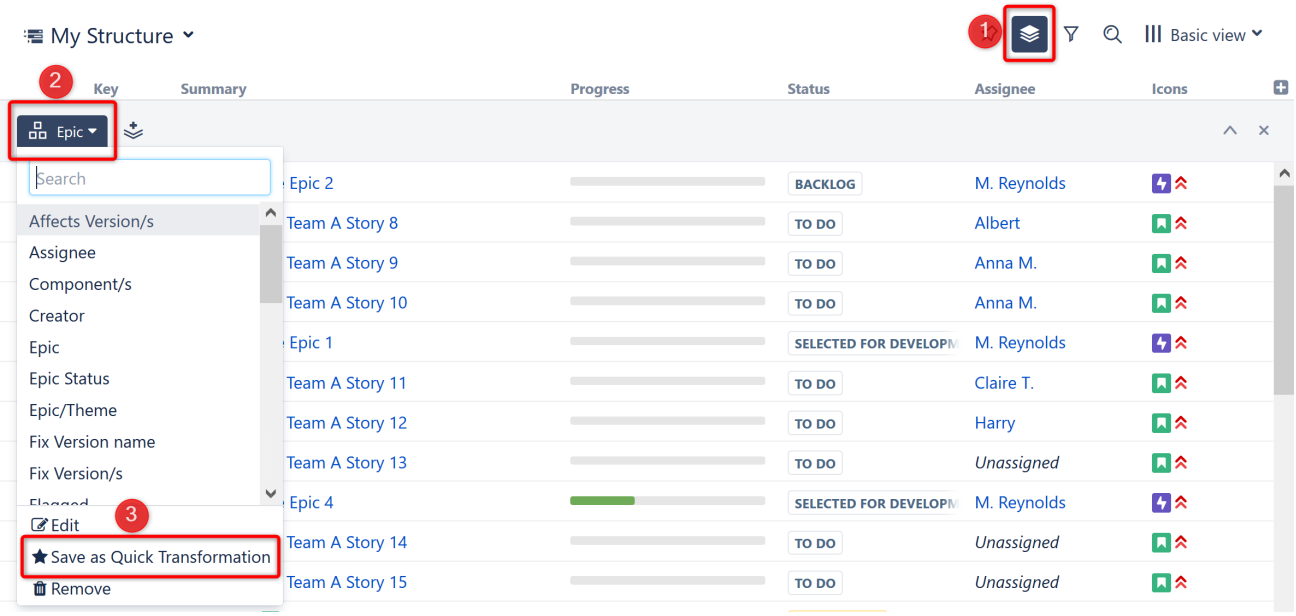
You can use keyboard shortcuts to toggle quick transformations, based on their position in the transformations list. The shortcut is **Q** and then the number (**1—9**), typed in quick succession.

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(see page 185)
2. Create your transformation
3. Click **Save as Quick Transformation**

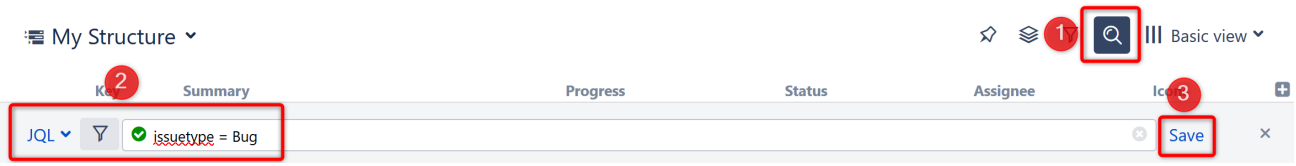


If you don't see a way to add a quick transformation, you likely do not have the appropriate permissions for that structure. You should ask the structure's owner to create the quick transformation.

Adding a Quick Filter

Filter quick transformations can also be created from a search:

1. Open Search(see page 180)
2. Enter your query (using Text, JQL, or S-JQL search)
3. Click **Save**



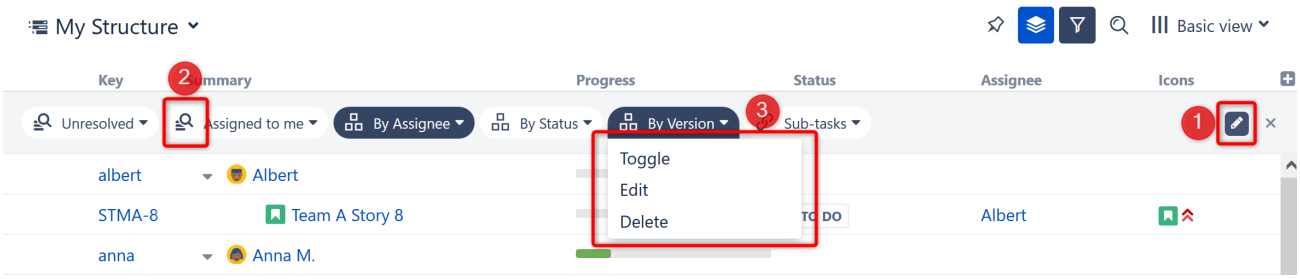
After clicking **Save**, you will be given the opportunity to name and configure the quick transformation.

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.

3.4.3.6 Default Quick Transformations

Default quick transformations are shown whenever a structure does not have customized quick transformations, or when displaying query results, clipboard items or other non-structure content.

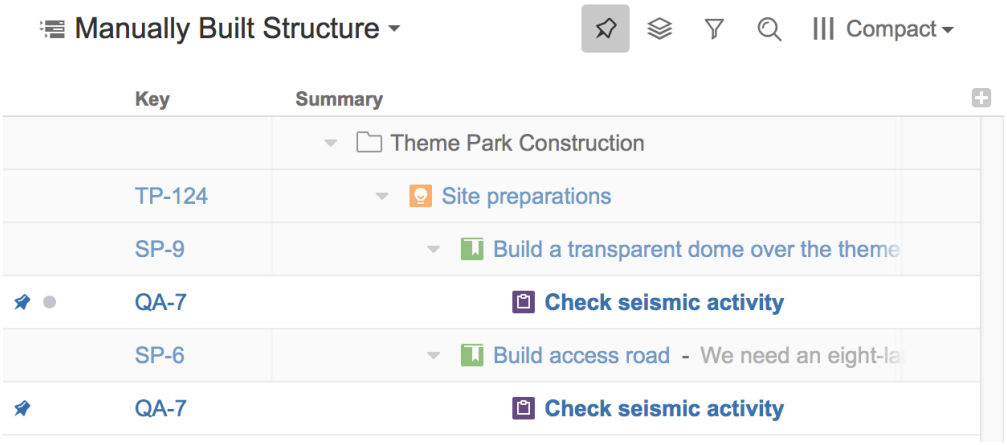
The following transformations are available:

Transformation	Effect of applying this transformation
Unresolved	Only issues with an empty Resolution field are shown
Assigned to me	Only issues assigned to the current user are shown
By Assignee	All top-level issues are grouped by Assignee
By Status	All top-level issues are grouped by Status
By Version	All top-level issues are grouped by Version
Sub-tasks	Sub-tasks are added to the structure under their parent tasks

Default quick transformations can be edited or deleted.

3.4.4 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.



Structure Panel on the Issue Page(see page 85) automatically pins the issue being displayed, so you only see the relevant part(s) of the structure.

3.4.4.1 What is Displayed in Pinned Item Mode?

When a structure is in Pinned Item Mode, only the following items are displayed:

- The pinned item itself (all instances of the item within the structure)
- All parent items of the pinned item, up to the top-level item
- All sub-items of the pinned item, down to the deepest level

Items that are "siblings" or located somewhere else in the hierarchy are not displayed.

3.4.4.2 Turning Pinned Item Mode On and Off

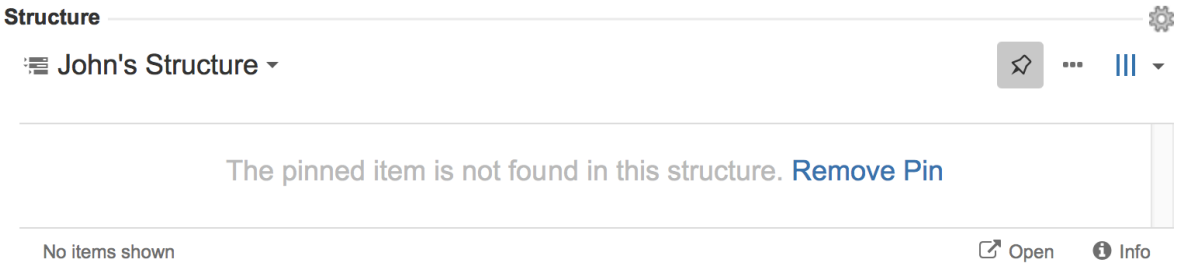
To turn Pinned Item Mode on or off, click the Pin button on the panel toolbar or press **Ctrl+.** on your keyboard.

When Pinned Item Mode is turned on:

- On Structure Board - whatever item is selected in the structure will be pinned. You can pin any issue on the Structure Board.
- On a Project Page or Agile Board - whatever issue is selected in the project/board's issue list will be pinned. To pin a new item, select a different item in the issue list.
- On an Issue Page - the corresponding issue will always be pinned. It it not possible to pin any other item when viewing a structure on an issue page.

Pinned Item Not Found

If the pinned item is not in the selected structure, you will receive the following message:



If you see this message you can:

- Click **Remove Pin** or click the **Pin** button to view the entire structure. If you want to add the selected issue to the structure, click the **Paste** button in the toolbar.
- Click the structure's name and select a new structure.
- If you are viewing the structure from a Project Page or Agile Board, select a new issue from the issue list.

3.4.4.3 Limitations Imposed by Pinned Item Mode

When an item is pinned, there are some restrictions to how items can be manipulated within the structure:

- Items above the pinned item (its parent/ancestors) cannot be moved or deleted
- You cannot add items above the pinned item or as siblings to the pinned item
- Items beneath the pinned item can be moved or deleted, and new items can be added below the pinned item

> Even though you can't move parent items while in Pinned Item Mode, you still can select them, edit or apply Jira operations.

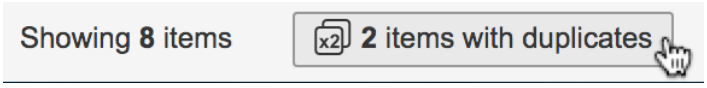
3.4.5 Identifying Duplicate Items

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.

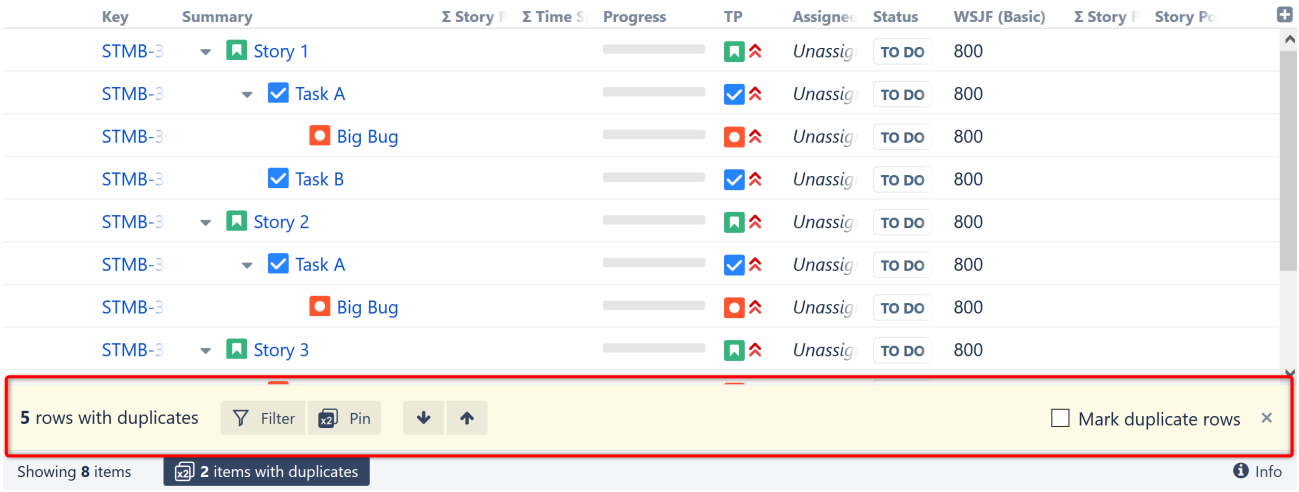
3.4.5.1 Finding Duplicates

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



> The duplicates counter works for the visible part of a structure only. If you apply filters on top of the existing structure (for example, transformations), and these filters hide all but one of a duplicated item, that item will no longer be counted.

Clicking the duplicates counter will open the Duplicates panel, which allows you to highlight, filter, pin or quickly navigate between duplicate items.



Duplicate Counts

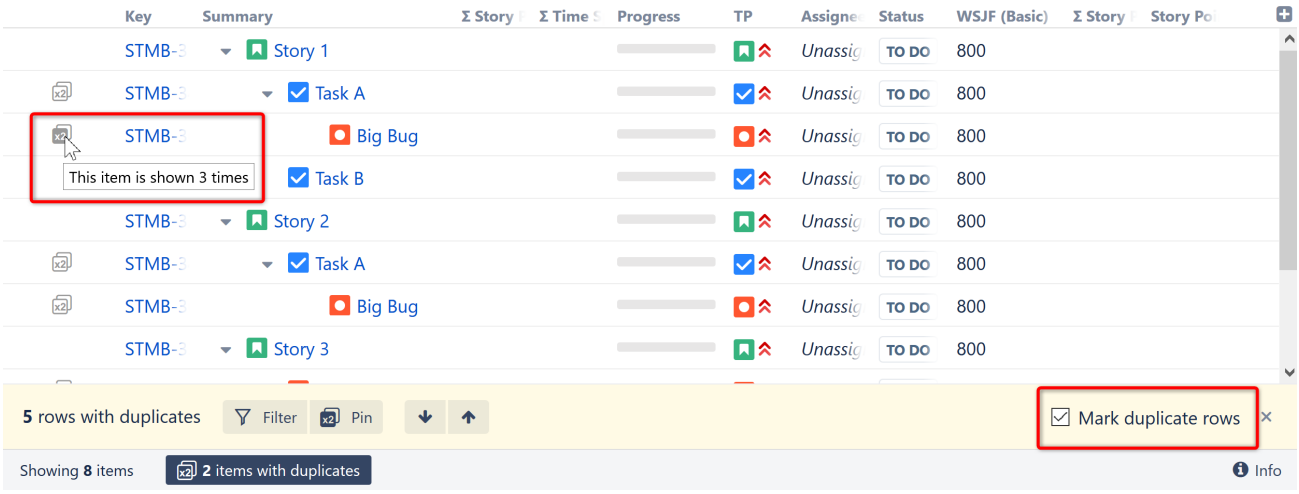
The duplicates counter shows the number of items that have duplicates. In the example above, there are two duplicating items (Task A and Big Bug).

The Duplicates panel shows the number of rows those items produce. In the example above, there are five rows of duplicates:

- Task A appears twice
- Big Bug appears three times

3.4.5.2 Mark Duplicate Rows

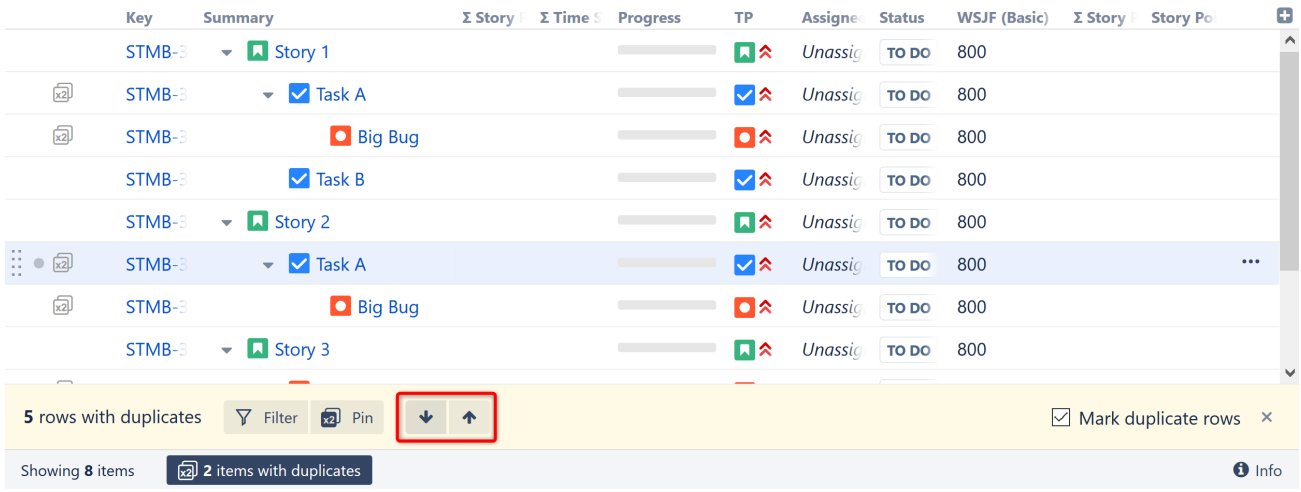
To highlight duplicate items, select the 'Mark duplicate rows' option. This places a duplicate icon (with ‘x2’ written on it) to the right of each duplicate row. If you hover over that icon, you'll see how many instances of this item are shown in the structure.



Once you highlight duplicate items, the highlighting stays even if you close the Duplicates Panel. To remove the highlighting, open the panel and clear the 'Mark duplicate rows' checkbox.

Navigating Between Duplicates

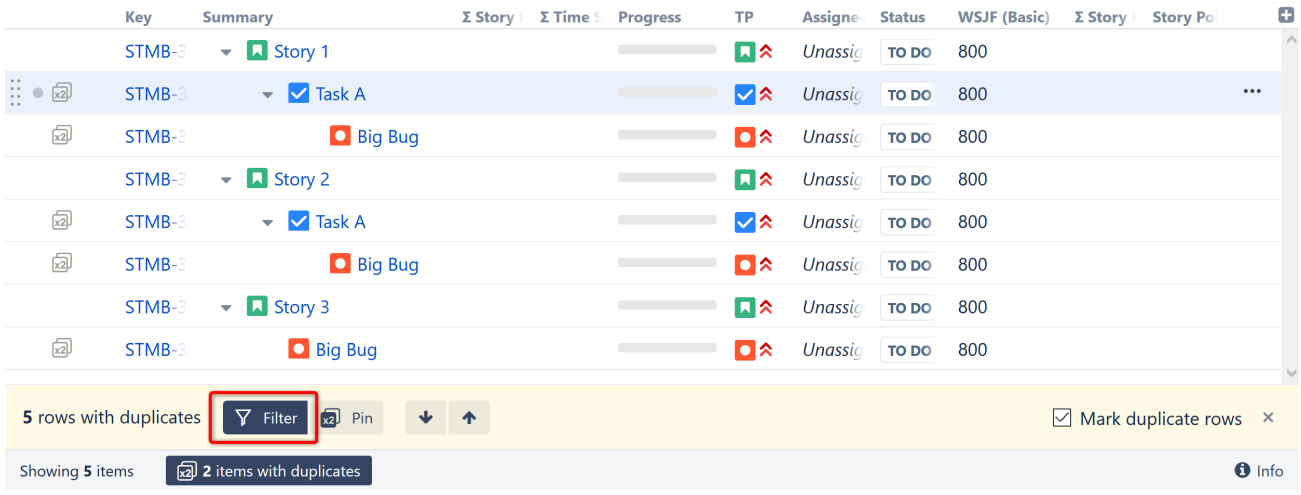
Clicking the “Up” and “Down” arrow buttons in the Duplicates panel moves the focus to the next/previous duplicate row.



When you navigate to a duplicate row that is in a collapsed part of a structure, this part of the structure will be expanded so the duplicate is visible.

3.4.5.3 Filtering by Duplicates

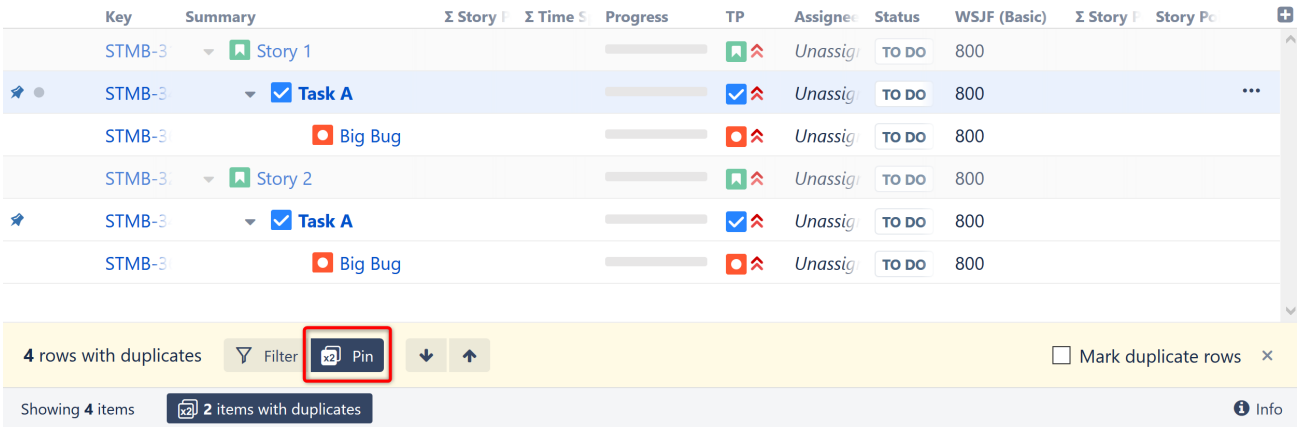
Clicking the Filter button hides everything in the structure except duplicate items and their parents.



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

3.4.5.4 Pinning Duplicates

The Pin button in the Duplicates panel hides everything except the selected item, its parents and its children. If the pinned item is a duplicate, all instances of that item and its parents/children will be displayed.



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

3.5 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(see page 207)

The following articles will show you how to create your own custom formulas or add one of our predefined formulas(see page 219) to a structure.

- Formula Columns(see page 197)
  - Columns as Variables(see page 203)
  - Formulas as Variables(see page 205)
  - Query Match as Variables(see page 206)
  - Work Time in Formula Columns(see page 206)
- Wiki Markup in Formula Columns(see page 207)
  - Wiki Markup Advanced Examples(see page 211)
- Bundled Formulas(see page 219)
- Expr Language(see page 221)
  - Expr Function Reference(see page 226)
  - Aggregate Function Reference(see page 246)
  - Expr Language Reference(see page 255)
  - Expr Pattern Matching(see page 263)
  - Expr Error Codes(see page 265)

3.5.1 Formula Columns

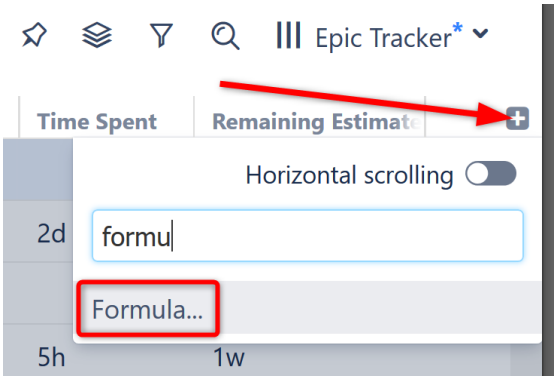
The following article will walk you step-by-step through the creation of a basic formula(see page 197).

- Add a Formula Column(see page 198)
- Formula(see page 198)
- Variables(see page 201)
- Options (Aggregation)(see page 201)

- Format(see page 202)
- Additional Information(see page 203)

3.5.1.1 Add a Formula Column

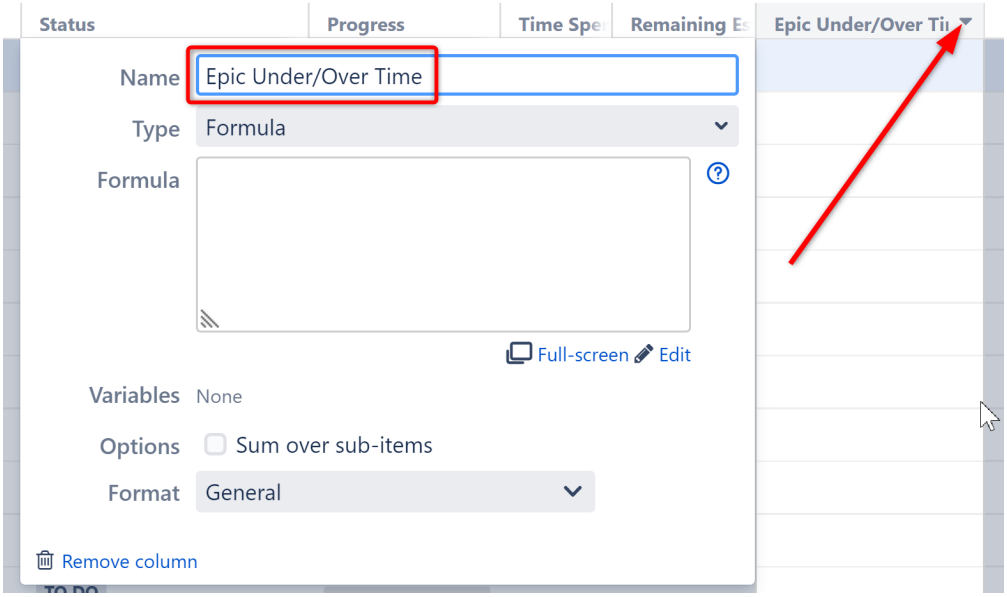
Start with adding a new column[22] by clicking the + icon to the right of the column headings. Select **Formula...** as its type.



Name Your Formula

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

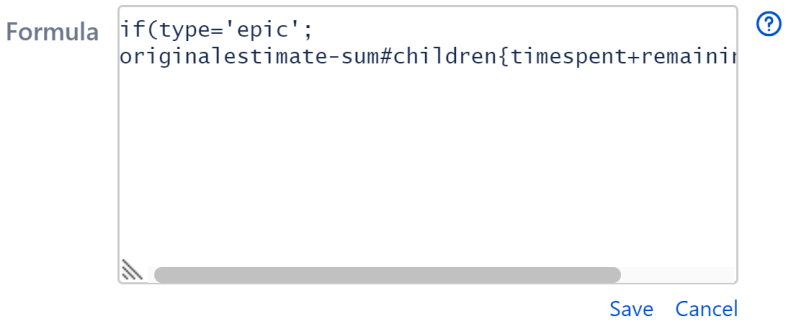
In the **Name** field, give the column a meaningful name – something that expresses the purpose of your formula and will be easily recognized by yourself and anyone you might share it with.



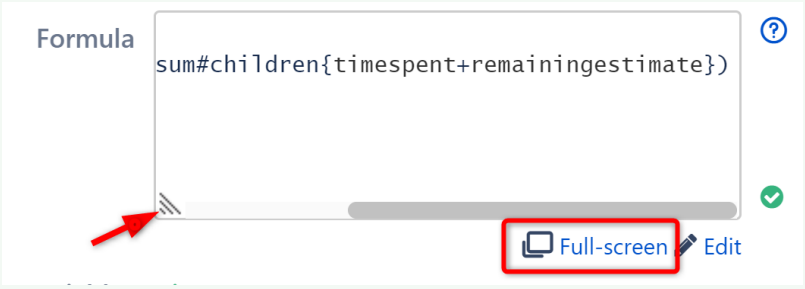
3.5.1.2 Formula

Enter your formula into the **Formula** field and click **Save**.

22 https://wiki.almworks.com/display/structure2gmaster/Customizing+Columns



Not enough space? Drag the corner of the field or click **Full-Screen** to expand the Formula window.



Formulas should be constructed using the Expr Language(see page 221), an intuitive language that supports variables, arithmetic operations and functions. You can find tutorials, reference guides, and several examples in our Expr Language documentation(see page 221).

The example above is a simple formula to calculate whether or not we're on target to complete each epic on time:

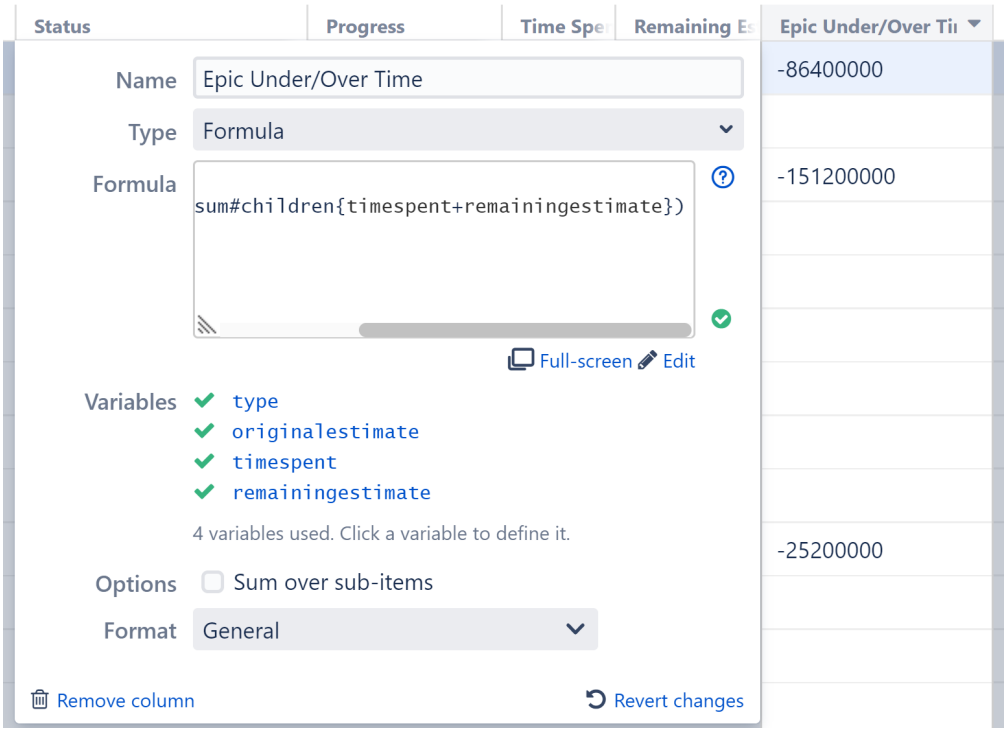
```
if(type='epic';
originalestimate-sum#children{timespent+remainingestimate})
```

In case you're not yet fluent in Expr(see page 221), 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.



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(see page 221) 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(see page 226) 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(see page 201).

Handling Unexpected Results

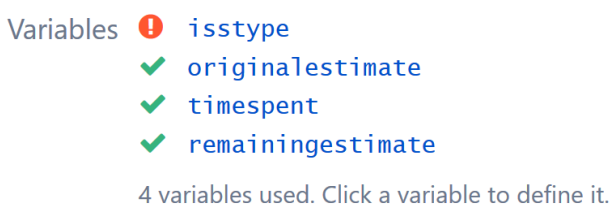
In some cases, the formula may pass inspection, but the results aren't what you expected. You may simply need to edit your Variables(see page 201), Options(see page 201) or Format(see page 202); or you may need to revise the formula itself by clicking the **Edit** button.

> 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.

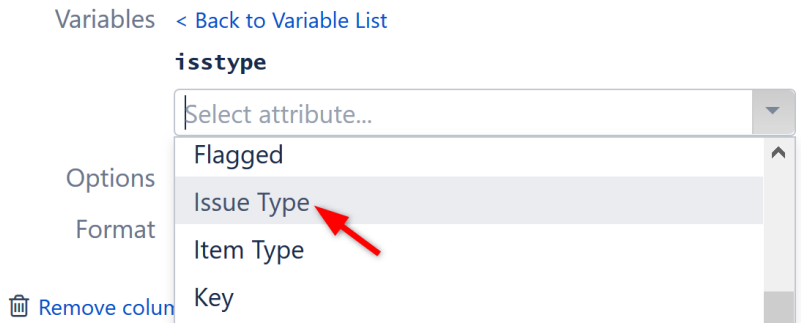
3.5.1.3 Variables

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(see page 203) or even another formula(see page 205).

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.



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.



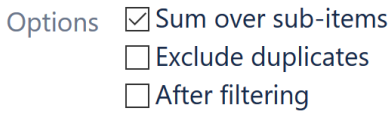
> The following names are automatically recognized by Structure:
>
> - Names of standard Jira fields, such as **Summary** or **Priority**.
> - Names of custom fields, with all non-letters removed and all spaces converted to underscores. For example, **Story_Points**.
> - Names starting with **Total_** or **Sum_** and having a well-known name afterwards, such as **Sum_Story_Points** or **Total_Estimate**. These are converted to a Sum(see page 246) 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(see page 203).

3.5.1.4 Options (Aggregation)

Select **Sum over sub-items** to have each row display an **aggregate total**, meaning the results for each row will be calculated as a sum of the values for that row and its sub-items.



When aggregation is enabled, you have a couple of options:

- **Exclude duplicates** - If an item appears more than once in the structure, it's value will only be included once within the aggregate total.
- **After filtering** - When checked, filtered items 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(see page 246) 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.

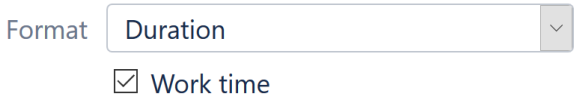
However, Structure has no way of knowing what each value represents, so these options are always available. When selecting this option, be careful to verify that the calculated values will make sense.

3.5.1.5 Format

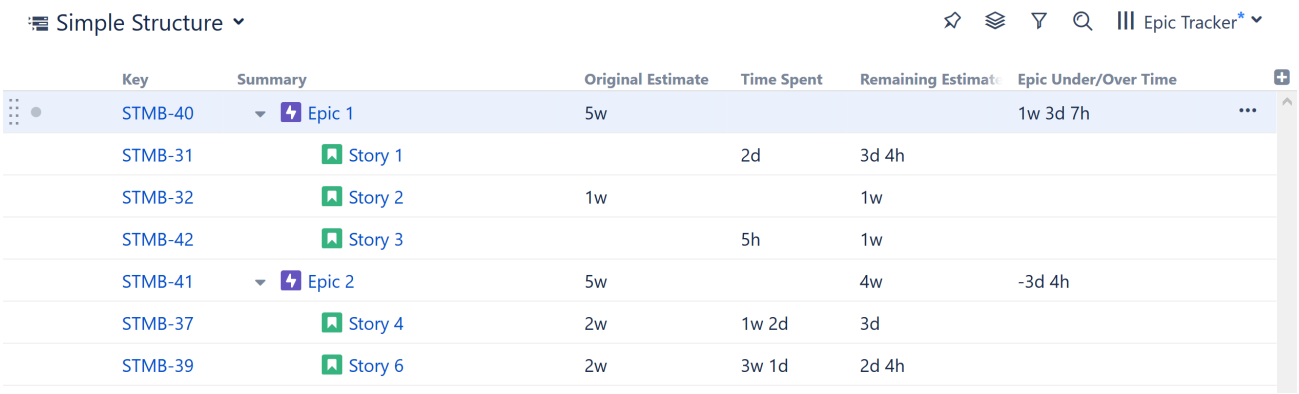
The **Format** section allows you to customize the format of your results. The following options are available:

- **General** - this default option will work for most formulas. If your results don't look right, try one of the others.
- **Number -** lets you specify the number of decimal places that will always be shown. The value will be rounded up to the least meaningful digit in this format.
- **Percentage** - treats the value as a ratio (0.0 = 0%, 1.0 = 100%) and adds a percent sign.
- **Date/Time** - displays the results as date/time and allows you to pick the appropriate format.
- **Duration** - displays duration values as days, hours and minutes. You can also select **Work time** to display values using Jira's time tracking settings, so the duration reflects your work hours. See Work Time in Formula Columns(see page 206) for more information.
- **Wiki Markup** - allows you to add wiki markup, including colors and images, to a column. See Wiki Markup in Formula Columns(see page 207) for more details.

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



Our new column (Epic Under/Over Time) now displays the weeks, days, and hours that we are either ahead of schedule or behind schedule for each epic:



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.

3.5.1.6 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(see page 339), which other users can select
- Creating a perspective URL(see page 92) that will open the structure with the same configuration, including the formula column

Sorting by Calculated Value

You can sort(see page 69) by the values calculated in a formula column by clicking the column header.

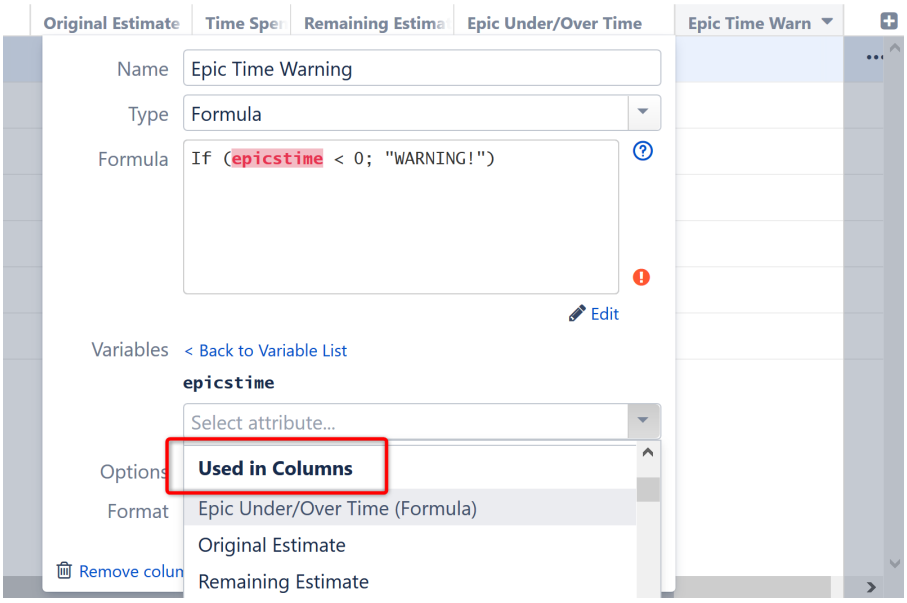
See Also

- Creating an Advanced Formula Column(see page 446)
- Bundled Formulas(see page 219)
- Expr Language(see page 221)

3.5.1.7 Columns as Variables

When using a formula(see page 197), 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.



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(see page 197). If the original formula resulted in a negative value, this column will list a simple "WARNING" text flag. (You could also make this flag more effective with Wiki Markup(see page 207)!)

Simple Structure

Simple Structure

Epic Tracker*

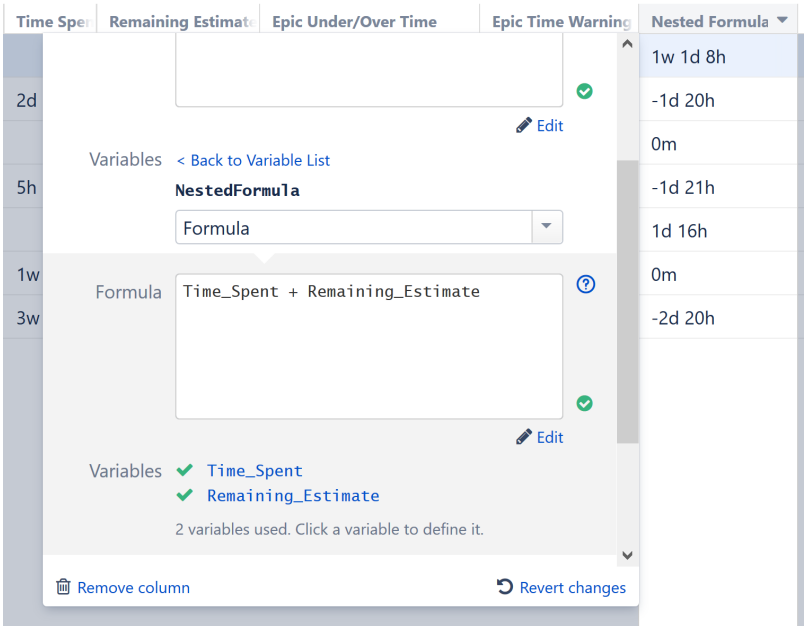
Key	Summary	Original Estimate	Time Spe	Remaining Estima	Epic Under/Over Time	Epic Time Warning
STMB-40	Epic 1	5w			1w 3d 7h	
STMB-31	Story 1		2d	3d 4h		
STMB-32	Story 2	1w		1w		
STMB-42	Story 3		5h	1w		
STMB-41	Epic 2	5w		4w	-3d 4h	WARNING!
STMB-37	Story 4	2w	1w 2d	3d		
STMB-39	Story 6	2w	3w 1d	2d 4h		

Once we've created 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.

3.5.1.8 Formulas as Variables

When using a formula(see page 197), you can assign another formula as a variable. This is similar to using columns as variables(see page 203), 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

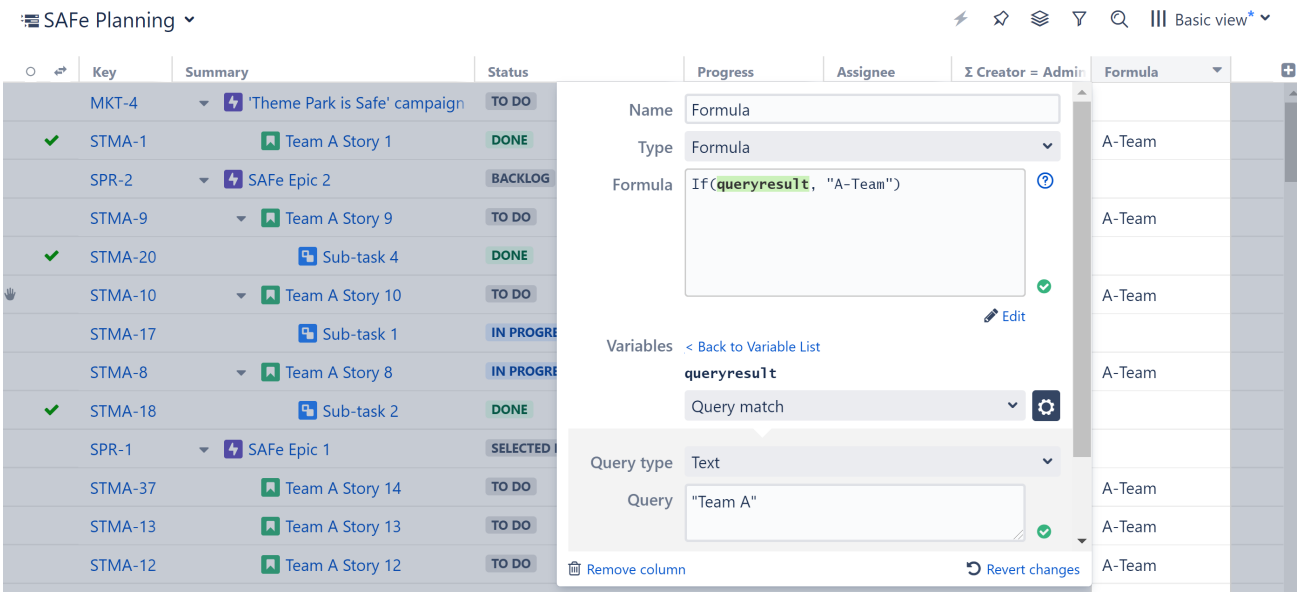
Nested formulas can also use formulas as variables. Doing this, you can create very complicated formulas that rely on several levels of nested formulas.

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.

3.5.1.9 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(see page 266) query.



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(see page 266) queries.
4. Enter the query.

> If you've already created a Query Match column(see page 328), you can also use the existing column as a variable(see page 203).

3.5.1.10 Work Time in Formula Columns

When the format of a formula column(see page 197) is set to **Work Time**, Structure uses Jira's time tracking settings to convert the number of hours to the number of days and weeks. By default, Jira is configured for an 8 hour work day, with 5 work days per week.

> If Duration is selected, but Work Time is not checked, hours are converted to days and weeks on a calendar basis.

Whether you need to use the **Work Time** option depends on where the value is coming from:

- When working with specific dates, you will probably want to keep the Work Time option off and see the calendar duration. For example, if you want to calculate the number of days a ticket remains open (`now() - created`), leaving the Work Time option will give you a more accurate result.
- When working with values retrieved from an issue's Original Estimate, Remaining Estimate and Time Spent fields, you will probably want to use Work Time option. For example, to calculate overspending (`time_spen`

`t + remaining_estimate - original_estimate`), selecting the Work Time option will give a result based on actual work hours.

3.5.2 Wiki Markup in Formula Columns

Customize your structure, call attention to critical information or color-code data fields using wiki markup within formula columns(see page 197).

Wiki markup allows you to:

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

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

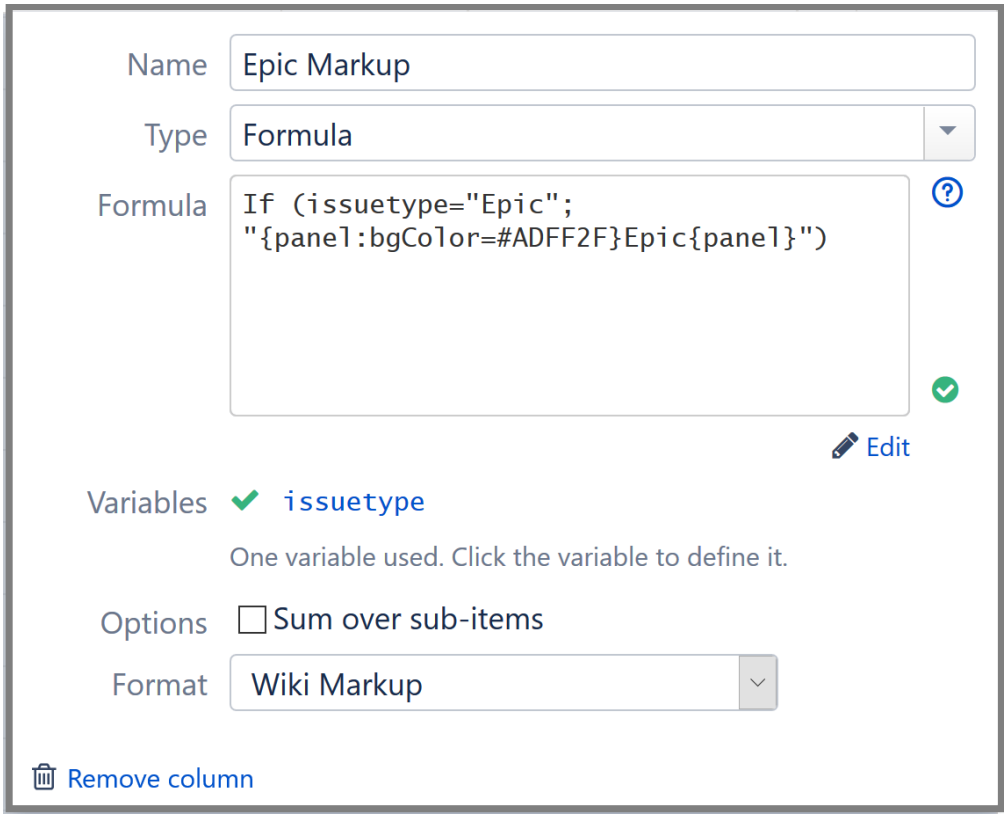
https://www.youtube.com/watch?v=7boTHUDge0c

3.5.2.1 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(see page 209) 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.)



As you save/update the formula, your new column should update automatically. Once you're finished, click anywhere on your structure to close the Add Column dialogue and see your new column.

The example above highlights all the Epics in the structure:

Here's the formula we used - just in case you want to try it yourself: **If (issuetype="Epic"; "{panel:bgColor=#ADFF2F}Epic{panel}")**

With a few more If statements, you could color-code your entire structure by issue type. Or you could assign different colors to each Assignee or some other custom field. The possibilities are endless!

3.5.2.2 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[23].

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.

3.5.2.3 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.

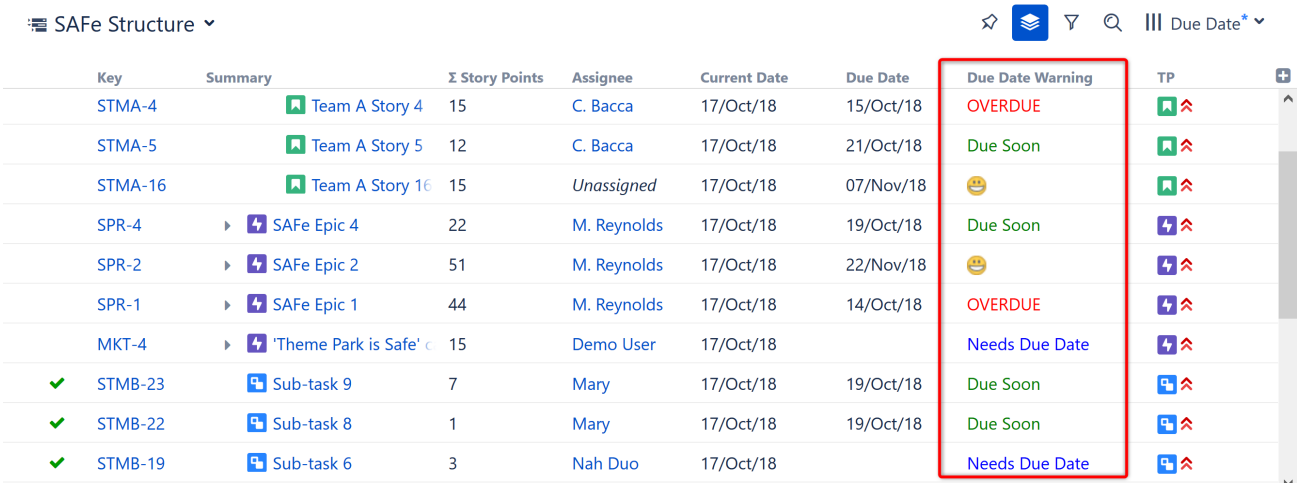
3.5.2.4 Examples

Example 1: Progress Warnings

In the following example, we have created a simple formula to draw attention to overdue and upcoming due dates:

- When an issue is overdue, a red "OVERDUE" warning appears in the column
- When an issue is due within the next 7 days, the columns displays a green “Due Soon”
- When there's over a week to go, the issue gets a smiley face
- And if the issue doesn't have a due date, it let's you know that too

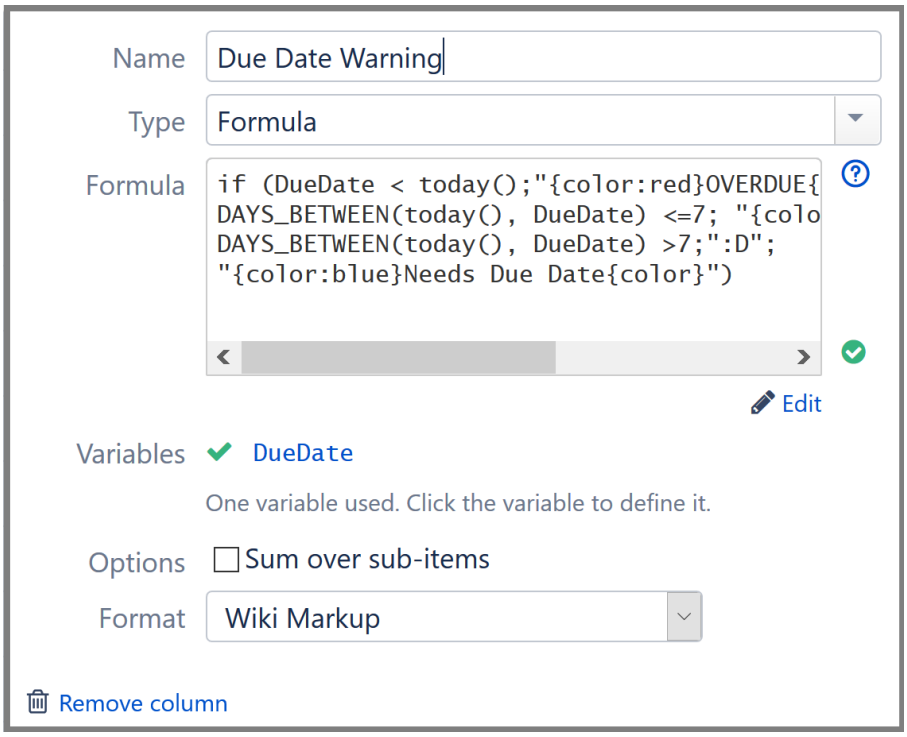
23 https://jira.atlassian.com/secure/WikiRendererHelpAction.jspa?section=all



Key	Summary	Σ Story Points	Assignee	Current Date	Due Date	Due Date Warning	TP
STMA-4	Team A Story 4	15	C. Bacca	17/Oct/18	15/Oct/18	OVERDUE	
STMA-5	Team A Story 5	12	C. Bacca	17/Oct/18	21/Oct/18	Due Soon	
STMA-16	Team A Story 16	15	*Unassigned*	17/Oct/18	07/Nov/18	😀	
SPR-4	SAFe Epic 4	22	M. Reynolds	17/Oct/18	19/Oct/18	Due Soon	
SPR-2	SAFe Epic 2	51	M. Reynolds	17/Oct/18	22/Nov/18	😀	
SPR-1	SAFe Epic 1	44	M. Reynolds	17/Oct/18	14/Oct/18	OVERDUE	
MKT-4	'Theme Park is Safe'	15	Demo User	17/Oct/18		Needs Due Date	
STMB-23	Sub-task 9	7	Mary	17/Oct/18	19/Oct/18	Due Soon	
STMB-22	Sub-task 8	1	Mary	17/Oct/18	19/Oct/18	Due Soon	
STMB-19	Sub-task 6	3	Nah Duo	17/Oct/18		Needs Due Date	

To accomplish this, we added markup language to a standard **If** statement:

if (DueDate < today(); "{color:red}OVERDUE{color}"; DAYS_BETWEEN(today(), DueDate) <=7; "{color:green}Due Soon{color}"; DAYS_BETWEEN(today(), DueDate) >7; ":D"; "{color:blue}Needs Due Date{color}")

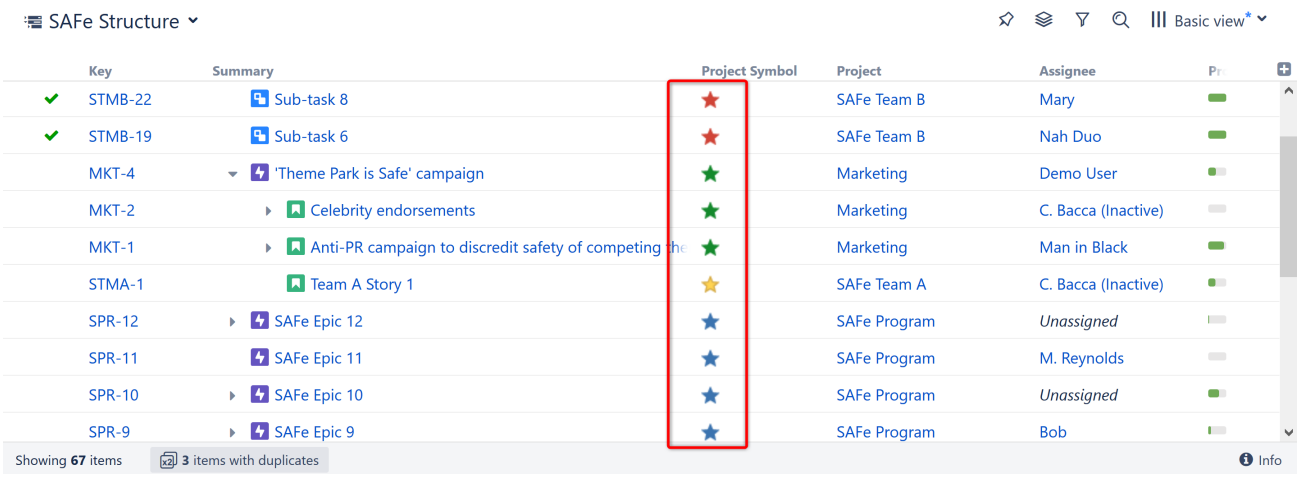


We used text to call attention to overdue items, but you could also add a flag: "(flag)"

To learn more about using If statements, DAY_BETWEEN, or any other functions, see Expr Function Reference(see page 226).

Example 2: Project Markers

In this example, we've created a column to quickly identify each project we're working on. In this case, each project is marked by a unique star color.



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

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!

3.5.2.5 Wiki Markup Advanced Examples

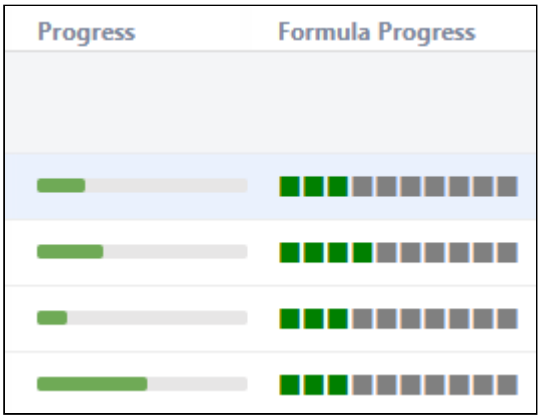
While wiki markup has its limitations, you can get pretty creative and visualize more complex metrics in Structure columns, such as custom progress bars, bar charts and much more. The use of colors and images greatly expands the possibilities.

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

- Customizable Progress Bar(see page 211)
- Customizable Status Bars(see page 212)
  - Multi-bar(see page 213)
  - Multi-bar with Image(see page 214)
  - Multi-bar with Numbers(see page 215)
- Simple Burn-down Chart(see page 216)

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}")
```

This can be especially useful if you want to display progress based on some complex fields, like a ScriptRunner scripted field, which is not supported by the standard formula column at the moment.

Customizable Status Bars

Wiki markup can also be used to create more complex progress calculations, based on multiple issue statuses.

In the following example, we created multiple custom status bars, tracking the following statuses:

- To Do = Red
- In Progress = Orange
- Done = Green
- All Other Statuses = Gray

As with our custom progress bar, these formulas can easily be modified to adjust status colors, include additional statuses or represent each status in a different format.

Multi-bar

We used the following code to build the Multi-bar Status Bar.

Multi-tiered progress bar

```
//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

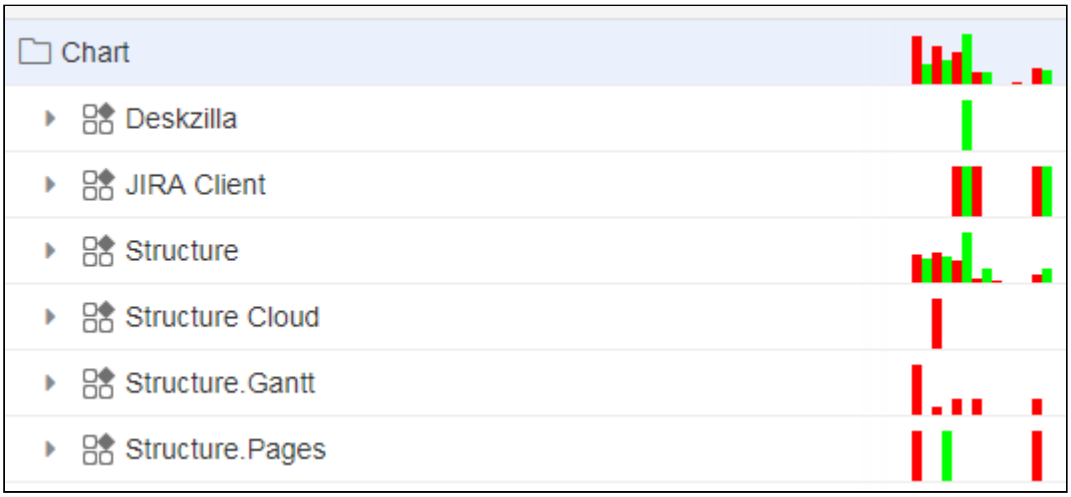
with all=COUNT{1}: with todoN=COUNT#truthy{status="to do"}: with todo=FLOOR(todoN/
all*granularity): with inprogressN=COUNT#truthy{status="in progress"}: with
inprogress=FLOOR(inprogressN/all*granularity): with doneN=COUNT#truthy{status="done"}
: with done=FLOOR(doneN/all*granularity): with otherN=all-todoN-inprogressN-doneN: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

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!",
```

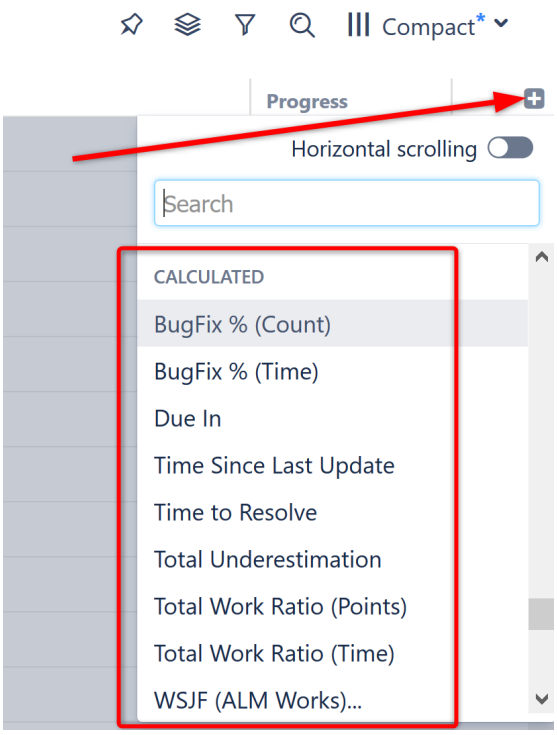
```
"!https://www.example.com/images/Red.png|height=", FLOOR(day5_created/maxth*heighth),
",width=5!",
"!https://www.example.com/images/Green.png|height=", FLOOR(day5_resolved/
maxth*heighth), ",width=5!",
"!https://www.example.com/images/Red.png|height=", FLOOR(day6_created/maxth*heighth),
",width=5!",
"!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.

3.5.3 Bundled Formulas

The simplest way to include formulas in a structure is to use one of our bundled formulas.(see page 331)

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)	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 SUM function(see page 246) to calculate a value for the issue and all its sub-issues.
Total Work Ratio (Points)	Ratio of total work done to the total amount of work. The amount of work is counted in Story Points, and issues are considered "done" when they have a non-empty Resolution field. This "Totals" formula uses the SUM function(see page 246) to calculate a value for the issue and all its sub-issues.
Total Work Ratio (Time)	Ratio of total work done to the total amount of work. The amount of work is based on the sum of Time Spent and Remaining Estimate values. This "Totals" formula uses the SUM function(see page 246) 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: • Job Size • User/Business Value • Time Criticality • Risk Reduction • Opportunity Enablement If you have such fields but they are not numeric (for example, a select list), edit the formula and replace the usage of a variable with a CASE() function, where you can assign individual numerical weights to each option.

Column Name	Description
WSJF (ALM Works)	Weighted Shortest Job First metric, according to categories used at ALM Works: • Benefit • Pain • Marketability • Impact • Cost • Risk • Clarity To use this formula, you must set up such fields with the following values: Nil, Low, Medium and High.
Σ Assignee Cost	Calculates the dollar amount of the task, based by the time (Time Spent + Remaining Estimate) multiplied by the per hour rate for the current Assignee. The rate is taken from the "Hourly Rate" additional property for the user who is the assignee. Shows the total amount for the issue and its sub-issues.

3.5.4 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(see page 197), 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.

- Language Components(see page 221)
- Basic Constructs (see page 222)
  - Variables(see page 222)
  - Functions(see page 222)
  - Numbers and Text Strings(see page 223)
  - Operations(see page 224)
- Advanced Constructs(see page 224)
  - Aggregate Functions(see page 224)
  - Local Variables(see page 225)
  - Comments(see page 225)

For a more in-depth study, see our Expr Reference Guides(see page 226).

You can view examples of Expr formulas by adding bundled formulas(see page 219) to your structure. To see the formula, simply open the column options(see page 331) panel.

3.5.4.1 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(see page 203) or even another formula(see page 205).
- 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.

3.5.4.2 Basic Constructs

Variables

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

- Issue fields
- Progress
- User properties
- Item type(see page 94)
- Status category(see page 325)
- Another column(see page 203)
- Another formula(see page 205)
- Query Match(see page 206)
- Last Comment(see page 330)

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(see page 197) 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(see page 226) 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(see page 255).

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.

3.5.4.3 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(see page 246) 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")
```

3.5.4.4 Expr Reference Guides

- Expr Function Reference(see page 226)
- Aggregate Function Reference(see page 246)
- Expr Language Reference(see page 255)
- Expr Pattern Matching(see page 263)
- Expr Error Codes(see page 265)

3.5.4.5 Expr Function Reference

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

- Notes About Functions(see page 228)
- Conditional Functions(see page 228)
  - CASE(see page 228)
  - CHOOSE(see page 228)
  - DEFINED(see page 229)
  - DEFAULT(see page 229)
  - IF(see page 229)
  - IFERR(see page 229)
  - ISERR(see page 229)
- Numeric Functions(see page 230)
  - ABS(see page 230)
  - CEILING(see page 230)
  - FLOOR(see page 230)
  - MAX(see page 231)
  - MIN(see page 231)
  - MOD(see page 231)
  - POW(see page 231)
  - ROUND(see page 231)
  - SIGN(see page 232)
  - SQR(see page 232)
  - SQRT(see page 232)
- Text Functions(see page 232)
  - CONCAT(see page 232)
  - EXACT(see page 233)
  - LEFT(see page 233)
  - LEN(see page 233)
  - LOWER(see page 233)

    - MATCH(see page 233)
    - MID(see page 234)
    - REPEAT(see page 234)
    - REPLACE(see page 234)
    - REPLACE_AT(see page 234)
    - RIGHT(see page 235)
    - SEARCH(see page 235)
    - SUBSTRING(see page 235)
    - TRIM(see page 236)
    - UPPER(see page 236)
- Date and Time Functions(see page 236)
    - DATE(see page 236)
    - DATE_ADD(see page 236)
    - DATE_SET(see page 237)
    - DATE_SUBTRACT(see page 237)
    - DAY(see page 238)
    - DAYS_BETWEEN(see page 238)
    - DATETIME(see page 238)
    - END_OF_MONTH(see page 239)
    - FORMAT_DATETIME(see page 239)
    - HOUR(see page 239)
    - HOURS_BETWEEN(see page 239)
    - MAKE_DATE(see page 240)
    - MAKE_DATETIME(see page 240)
    - MINUTE(see page 240)
    - MONTH(see page 240)
    - MONTHS_BETWEEN(see page 240)
    - NOW(see page 240)
    - PARSE_DATETIME(see page 241)
    - SECOND(see page 241)
    - START_OF_MONTH(see page 241)
    - TODAY(see page 241)
    - TRUNCATE_TIME(see page 242)
    - TRUNCATE_TO_HOURS(see page 242)
    - TRUNCATE_TO_MINUTES(see page 242)
    - TRUNCATE_TO_SECONDS(see page 242)
    - WEEKDAY(see page 242)
    - YEAR(see page 242)
    - YEARS_BETWEEN(see page 243)
- Duration Functions(see page 243)
    - CALENDAR_DAYS(see page 243)
    - CALENDAR_HOURS(see page 243)
    - CALENDAR_MINUTES(see page 243)
    - CALENDAR_SECONDS(see page 244)
    - DURATION(see page 244)
    - FORMAT_DURATION(see page 244)
    - JIRA_DAYS(see page 244)
    - JIRA_DURATION(see page 244)
    - JIRA_WEEKS(see page 245)
- User-related Functions(see page 245)
    - USER_NAME(see page 245)
    - USER_DISPLAY_NAME(see page 245)
    - USER_EMAIL(see page 245)

    - USER_IS_ACTIVE(see page 245)
- Miscellaneous Functions(see page 245)
    - ME(see page 245)
    - NUMBER(see page 246)
    - TEXT(see page 246)

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(see page 263) 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(see page 265) 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^{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^{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^{th} 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(see page 263) 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(see page 263) 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?"; "/[aeuio]/") → "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(see page 263) 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(see page 197) 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")`

The second parameter (Number) must be an integer.

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[24].
- `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`

---

24 https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html

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` occ urs 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[25].
- `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()`

25 https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html

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`

User-related Functions

USER_NAME

`USER_NAME(UserKey)`

Returns the username specified by the user key.

USER_DISPLAY_NAME

`USER_DISPLAY_NAME(UserKey)`

Returns the user's full name, defined by the user key.

USER_EMAIL

`USER_EMAIL(UserKey)`

Returns the user's email, defined by the user key.

USER_IS_ACTIVE

`USER_IS_ACTIVE(UserKey)`

Returns 1 for active user, 0 for inactive user.

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"`

3.5.4.6 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.

- Aggregation Functions(see page 247)
  - AVG(see page 247)
  - COUNT(see page 247)
  - JOIN(see page 248)
  - MAX(see page 248)
  - MIN(see page 248)
  - PARENT(see page 249)
  - SUM(see page 249)
- Aggregation Modifiers(see page 249)
  - #afterChildren(see page 249)
  - #all(see page 250)
  - #ancestors(see page 250)
  - #baseLevel(see page 250)
  - #beforeChildren(see page 250)
  - #children(see page 250)
  - #depth(see page 251)
  - #distinct(see page 251)
  - #fromDepth(see page 251)
  - #fromLevel(see page 251)
  - #leaves(see page 252)
  - #level(see page 252)
  - #levels(see page 253)
  - #preceding(see page 253)
  - #reverse(see page 253)
  - #separator(see page 254)
  - #strict(see page 254)

- #subtree(see page 254)
- #toDepth(see page 254)
- #toLevel(see page 254)
- #truthy(see page 255)

⚠ 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}
T1	3	2
T1.1	2	2
T1.2		1
T1.2.1	1	1

✓ If a certain issue (or another kind of item) is included multiple times in the sub-tree, the average value will include the value for that issue only *once*. This behavior can be overridden by using the #all modifier.

Accepts modifiers: #all(see page 250), #children(see page 250), #leaves(see page 252), #strict(see page 254), #subtree(see page 254), #preceding(see page 253), #levels(see page 253) (together with #preceding), #baseLevel(see page 250) (together with #preceding).

COUNT

Count calculates a count of defined values (or truthy values, if the #truthy modifier is specified) for the item and/or its sub-items.

Summary	X	COUNT{X}
T1	3	3
T1.1	2	1
T1.2		1
T1.2.1	1	1

✓ If a certain issue (or another kind of item) is included multiple times in the sub-tree, it will be counted only *once*. This behavior can be overridden by using the #all modifier.

Accepts modifiers: #all(see page 250), #children(see page 250), #leaves(see page 252), #strict(see page 254), #subtree(see page 254), #truthy(see page 255), #preceding(see page 253), #levels(see page 253) (together with #preceding), #baseLevel(see page 250) (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(see page 254) modifier is set, join appends the values for children, wrapping them into characters (braces by default).
- Wrapping characters can be set by #beforeChildren(see page 250) and #afterChildren(see page 249) (see example for #subtree(see page 254) to see how it works).

Summary	X	JOIN{X}
T1	3	3
T1.1	2	3, 2
T1.2		3, ?
T1.2.1	1	3, ?, 1

Accepts modifiers: #ancestors(see page 250), #subtree(see page 254), #children(see page 250), #leaves(see page 252), #strict(see page 254), #reverse(see page 253), #separator(see page 254), #beforeChildren(see page 250), #afterChildren(see page 249), #fromLevel(see page 251), #toLevel(see page 254), #distinct(see page 251).

MAX

Max returns the maximum defined value for the item and/or its sub-items. Numeric, date, duration and text fields can be compared. Text fields are compared lexicographically.

Summary	X	MAX{X}
T1	3	3
T1.1	2	2
T1.2		1
T1.2.1	1	1

Accepts modifiers: #children(see page 250), #leaves(see page 252), #strict(see page 254), #subtree(see page 254).

MIN

Min returns the minimum defined value for the item and/or its sub-items. Numeric, date, duration and text fields can be compared. Text fields are compared lexicographically.

Summary	X	MIN{X}
T1	3	1
T1.1	2	2
T1.2		1
T1.2.1	1	1

Accepts modifiers: #children(see page 250), #leaves(see page 252), #strict(see page 254), #subtree(see page 254).

PARENT

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

Summary	X	PARENT{X}
T1	3	
T1.1	2	3
T1.2		3
T1.2.1	1	

Accepts modifier: #level(see page 252).

SUM

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

Summary	X	SUM{X}
T1	3	6
T1.1	2	2
T1.2		1
T1.2.1	1	1

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

- `SUM{}` (the same as `SUM#subtree{}`) aggregates values from all descendants
- `SUM#children{}` aggregates values from direct children only
- `SUM#leaves{}` aggregates values from leaves
- `SUM#preceding{}` aggregates values from the preceding rows

Note that when the value of the expression under aggregation is not numeric (and cannot be converted(see page 223) to a number), it is ignored.

If a certain issue (or another kind of item) is included multiple times in the sub-tree, the sum will include the value for that issue only *once*. This behavior can be overridden by using the `#all` modifier.

Accepts modifiers: #all(see page 250), #children(see page 250), #leaves(see page 252), #preceding(see page 253), #strict(see page 254), #subtree(see page 254).

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

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

Can be used with: join(see page 248).

#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)
T1		2	6	1	3
T1.1	2	2	2	1	1
T1.1	2	2	2	1	1
T1.1	2	2	2	1	1

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

Can be used with: sum(see page 249), count(see page 247), avg(see page 247).

#ancestors

Only process ancestors of the current row. This is the default behavior for join(see page 248).

Can be used with: join(see page 248).

#baseLevel

Can be used with SUM(see page 249) with the `#preceding` modifier to specify at which level the accumulation will be reset and start over from zero. This allows for accruing independent cumulative values in different sub-trees.

- By default, baseLevel is 0, which means that the accumulation will never start over and will cover the whole structure.
- If baseLevel is not zero, then the accumulation will be restarted once it reaches a row at the base level or higher in the hierarchy. For example, if you have epics at the top level, and stories underneath them, `SUM#preceding#baseLevel=1{story_points}` will accumulate the Story Points within the scope of each epic independently.

Can be used with: avg(see page 247), count(see page 247), sum(see page 249) (together with #preceding).

#beforeChildren

See #afterChildren(see page 249).

#children

Only process direct children of the current row.

Example

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

Can be used with: sum(see page 249), count(see page 247), avg(see page 247), join(see page 248), min(see page 248), max(see page 248).

#depth

Same as #level(see page 252) modifier.

#distinct

Makes join(see page 248) only concatenate distinct values. A duplicate value won't be added more than once if this modifier is on.

Modifiers #beforeChildren(see page 250) and #afterChildren(see page 249) don't work when this option is on.

Example

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

Can be used with: join(see page 248).

#fromDepth

Same as #fromLevel(see page 251).

#fromLevel

Specifies the position of the first row the aggregate function should take as input for a sequence.

Position is specified by an integer parameter denoted as n below:

- Positive values mean the absolute depth of the row in the structure, e.g. n=1 means root.
- Negative values mean the depth relative to current row, e.g. n=-1 is the current item's direct parent.
- Default is 1.
- Zero means the "super-root" row, which is a fictional parent of all the top rows. It can be used to get the value of another aggregate function applied to the whole structure. For example, `JOIN#fromLevel=0{MIN{due_date}}` will provide a sequence of the earliest due dates, starting from the earliest throughout the whole structure, then the earliest throughout the root tree this item is in, and so on.

This modifier does not work with any tree types except #ancestors(see page 250).

Example

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

Can be used with: join(see page 248).

#leaves

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

Example

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

Can be used with: sum(see page 249), count(see page 247), avg(see page 247), join(see page 248), min(see page 248), max(see page 248).

#level

When used with PARENT(see page 249), specifies the position of the parent that possesses value.

Position is specified by an integer parameter denoted as n below:

- Positive values mean the absolute depth of the row in the structure: n=1 means root element, n=2 means an element at the 2nd level, etc.
- Negative values mean the depth relative to the current row: n=-1 is the current item's direct parent.
- Default is -1.
- Zero means the "super-root" row, which is a fictional parent of all the top rows. It can be used to get the value of another aggregate function applied to the whole structure. For example, `PARENT#level=0{SUM{story_points}}` means total story points for the whole structure (including subtrees for all roots).

When used with SUM(see page 249) with the `#preceding` modifier, this specifies the level at which the values should be aggregated.

Example

Summary	X	-1	-2	1	2
T1	3			3	
T1.1	2	3		3	2
T1.2		3		3	
T1.2.1	1		3	3	

```
PARENT#level=-1{X}   // default one
PARENT#level=-2{X}   // "grandparent"
PARENT#level=1 {X}   // root row
PARENT#level=2 {X}
```

Can be used with: parent(see page 249), sum(see page 249) (together with #preceding), count(see page 247) (together with #preceding), avg(see page 247) (together with #preceding).

#levels

Can be used with SUM(see page 249) with the `#preceding` modifier to specify at which levels should the accrual of the values happen.

- It can be a single numeric value, for example: `SUM#preceding#levels=1{story_points}` will accumulate Story Points from top to bottom at level 1.
- It can be a list of numbers, in which case the list must be wrapped in quotes: `SUM#preceding#levels="2,4"{time_spent}` will accumulate Time Spent on levels 2 and 4.
- By default, all levels are counted.

Note that if you use the #baseLevel(see page 250) modifier, only values at levels that are deeper than the base level will be counted.

You can also use #level(see page 252) instead of #levels.

Can be used with: avg(see page 247), count(see page 247), sum(see page 249) (together with #preceding).

#preceding

Can be used with SUM(see page 249) to calculate a numeric total of the current item and all items above it in the structure.

Can be combined with the following modifiers:

- #baseLevel(see page 250) - the sum restarts whenever the specified level is reached
- #levels(see page 253) - the sum will only include the levels specified
- #all(see page 250) - items that appear more than once will be counted multiple times

Example

Summary	X	SUM#preceding{X}	SUM#preceding#baseLevel=1{X}	SUM#preceding#levels="1,3"{X}
T1	1	1		1
T1.1	2	3	2	
T1.2	3	6	5	
T1.2.1	4	10	9	5
T1.2.2	3	13	12	8
T1.3	2	15	14	
T2	2	17		10
T2.1	3	20	3	
T2.2	2	22	5	

```
SUM#preceding{X}
SUM#preceding#baseLevel=1{X}
SUM#preceding#levels="1,3"{X}
```

Can be used with: sum(see page 249).

#reverse

Reverses the order of row processing.

Example

```
JOIN#reverse{X}
```

Can be used with: join(see page 248).

#separator

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

Example

```
JOIN#separator="->"{X}
```

Can be used with: join(see page 248).

#strict

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

Cannot be used together with #children(see page 250), #ancestors(see page 250) or #leaves(see page 252), since these already exclude the current row.

Example

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

Can be used with: sum(see page 249), count(see page 247), avg(see page 247), join(see page 248), min(see page 248), max(see page 248).

#subtree

Process the whole subtree of the current row. This is the default behavior for sum(see page 249), count(see page 247), avg(see page 247), min(see page 248), max(see page 248).

Example

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

Can be used with: sum(see page 249), count(see page 247), avg(see page 247), join(see page 248), min(see page 248), max(see page 248).

#toDepth

Same as #toLevel(see page 254).

#toLevel

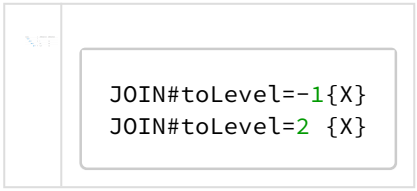
Specifies the position of the last row the aggregate function should take as input for a sequence.

Position is specified by an integer parameter denoted as n below:

- Positive values mean the absolute depth of row in the structure, e.g. n=1 means root.
- Negative values mean the depth relative to current row, e.g. n=-1 is the current item's direct parent.
- 0 means current row.
- Default is 0.

This modifier does not work with any tree types except #ancestors(see page 250).

Example



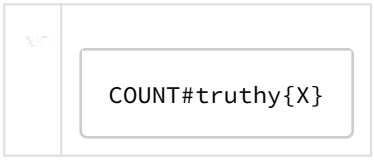
```
JOIN#toLevel=-1{X}
JOIN#toLevel=2 {X}
```

Can be used with: join(see page 248).

#truthy

Only count row if the subexpression produces a truthy value (see page 258).

Example



```
COUNT#truthy{X}
```

Can be used with: count(see page 247).

3.5.4.7 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(see page 256)
  - Comments(see page 256)
- Values(see page 256)
  - Undefined(see page 256)
  - Text(see page 256)
  - Numbers(see page 257)
  - Text to Number Conversion(see page 257)
  - Falsy and Truthy Values(see page 258)
- Variables and functions(see page 258)
  - Identifiers(see page 259)
  - Variables(see page 259)
  - Local Variables(see page 259)
  - Function Calls(see page 259)
  - Aggregate Function Calls(see page 259)
- Single-argument operators(see page 260)
  - NOT(see page 260)
  - + -(see page 260)

- Logical and arithmetic operators(see page 260)
  - Logical operators(see page 260)
  - Comparison operators(see page 261)
    - Equality: = (==).(see page 261)
    - Inequality: <> (!=)(see page 261)
    - Ordering(see page 261)
  - Arithmetic operators(see page 261)
- Precedence of operators(see page 262)
- Railroad diagrams(see page 262)
  - expression(see page 262)
  - logical-expression(see page 262)
  - arithmetic-expression(see page 263)
  - function-call(see page 263)
  - aggregation-expression(see page 263)

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.

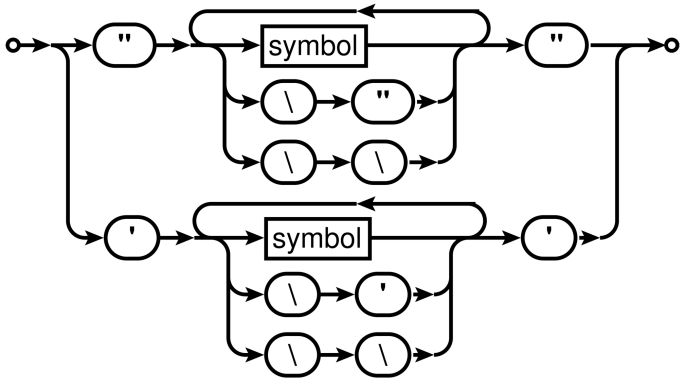
Functions can return this value when the result of the function is not specified. For example, the function `IF(N = 0; "No apples"; N = 1; "One apple")` only has a specified value when N is equal to 0 or 1. If N is equal to anything else, it returns `undefined`.

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(see page 197) 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(see page 226) 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[26]; 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.

---

26 https://confluence.atlassian.com/adminjiraserver071/choosing-a-default-language-802592304.html

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(see page 197) 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(see page 249).

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 *falsy* 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(see page 246) 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.

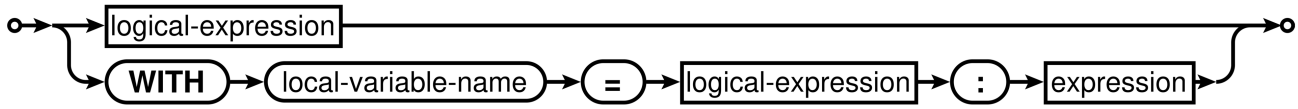
Single-argument operators are always evaluated first. Other operators in Expr language have the following precedence:

5 (highest)	`* /`
4	`+ -`
3	`= <> < > <= >=`
2	AND
1 (lowest)	OR

Railroad diagrams

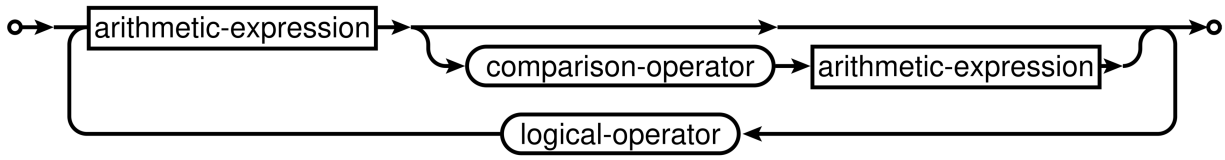
These diagrams display the complete syntax of Expr language.

expression



`local-variable-name` is an identifier(see page 259).

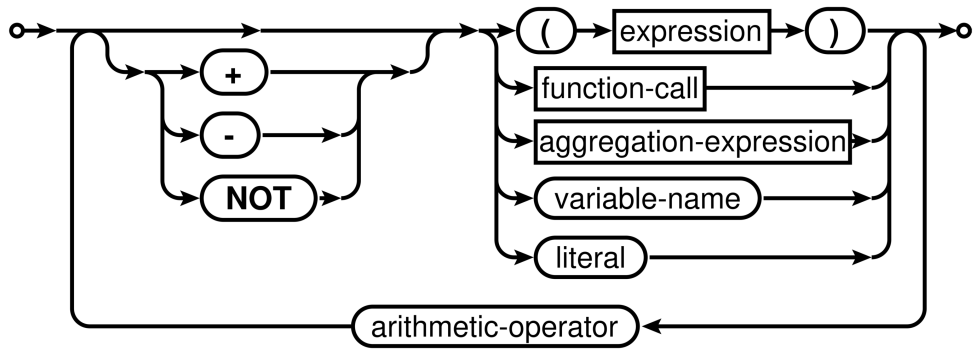
logical-expression



`comparison-operator` is one of these: `= <> < > <= >=`.

`logical-operator` is one of these: `AND OR`.

arithmetic-expression

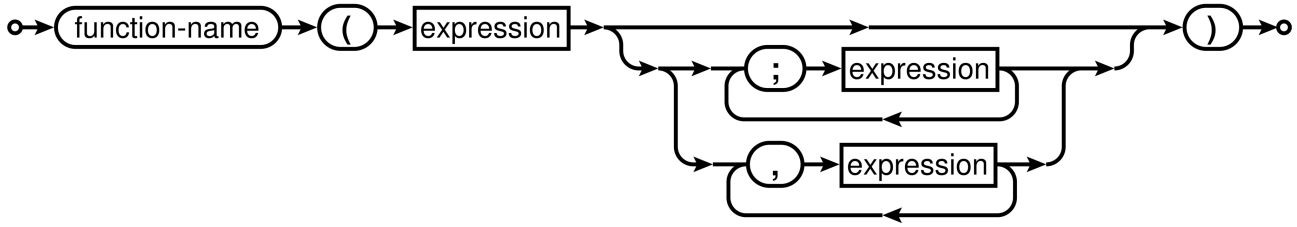


`variable-name` is an identifier(see page 259).

`literal` is either a number literal(see page 257), a text(see page 256) or UNDEFINED(see page 256).

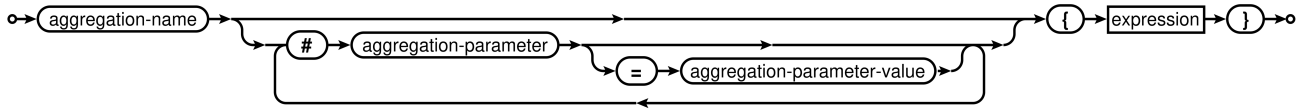
`arithmetic-operator` is one of these: `+ - * /`.

function-call



`function-name` is an identifier(see page 259).

aggregation-expression



`aggregation-name` and `aggregation-parameter` are identifiers(see page 259).

`aggregation-parameter-value` is either a text(see page 256) or a number literal(see page 257) with optional sign (either + or –).

3.5.4.8 Expr Pattern Matching

Expr language allows you to check text values against a certain pattern, when using the MATCH(see page 233), CASE(see page 228), REPLACE(see page 234) or SEARCH(see page 235) function.

There are three types of patterns that can used: Exact Matching(see page 264), Wildcard Matching(see page 264), and Regular Expression Matching(see page 264).

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[27].

The regular expression matching is different from other types of matching. The following rules apply:

- Leading and trailing whitespace characters are **not** removed.
- Text comparison is case-insensitive, like with the other types of matching.
- The value does not have to fully match the pattern – it is sufficient that at least one occurrence of the pattern is found in the value. To make your pattern match the whole text, use "^" and "$" characters in the pattern.

Regular expression matching is turned on if the first and the last characters of the pattern are "`/`". (These characters are removed, as they are not a part of the pattern.)

27 https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

3.5.4.9 Expr Error Codes

Evaluating Expr expressions may produce errors. Normally these errors are shown to the user with a human-readable message. However, in some cases you might need to check for a specific error using the ISERR(see page 229) 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. To fix, review your formula to ensure you're not dividing by zero. (To avoid division by zero, use the IF(see page 229) function.) If this does not work, try separately calculating parts of the formula to identify the error.
5	Variable Error	VAR!	An attribute that a variable was bound to produced an error. To fix, review the attributes bound to the expression variables and see why they could have produced an error for the row that shows this error.
6	Function Execution Error	FUNC!	A function suffered an internal error. Refer to logs for details.
7	Value Conversion Error	VALUE?	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.

Error Code	Name	Displayed As	Description
8	Malformed Regex	REGEX?	A text with invalid regular expression was passed to a matching function. Check the regular expressions that you use. See Expr Pattern Matching(see page 263) for details.
9	Internal Error	ERROR!	There's an internal problem with Jira or Structure. Refer to the logs for details.
10	Invalid Value	VALUE!	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	AGGR?	The formula contains an unknown aggregate function or an invalid aggregate function modifier. Check that your aggregate function and modifiers(see page 246) are valid and spelled correctly.

3.6 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(see page 293)
- in the Search Area on the Structure Widget — see Search(see page 180)
- in the S-JQL Filter generator(see page 130)
- in the workflow validator or condition — see Workflow Integration(see page 494)

3.6.1 Get Started with S-JQL

We recommend starting with the S-JQL Cookbook(see page 267), which contains a number of common examples that can be used as is or modified to meet your specific needs.

3.6.2 Master S-JQL

For a comprehensive description of the language and `structure()` JQL function, consult the S-JQL Reference(see page 272).

- S-JQL Cookbook(see page 267)
- S-JQL Reference(see page 272)
- structure() JQL function(see page 293)

3.6.3 S-JQL Cookbook

Here are the most common examples of using S-JQL.

- Find issues added to a structure(see page 267)
- Quick Filter for JIRA Agile's (GreenHopper) Scrum Board to display only low-level issues in a structure(see page 267)
- Retrieve all Epics in a certain status and all of their children(see page 268)
- Find Test Cases associated with Stories in an active sprint(see page 268)
- Find all issues that are blocking critical issues(see page 269)
- Find all unassigned issues in a part of a project(see page 269)
- Top-level view on unfinished parts of a project(see page 269)
- Find violations of the rule "Tasks must be under Epics or Stories"(see page 270)
- Find violations of the rule "An issue cannot be resolved if it has unresolved children"(see page 270)
- Find issues that can be resolved because all their children are resolved(see page 270)
- Get a view of a second (third, ...) level of the hierarchy(see page 271)
- Get the contents of a folder(see page 271)

3.6.3.1 Find issues added to a structure

Goal: Suppose that you are using a structure named "My todo list" as a collection of issues, and you want to see in the Issue Navigator all issues added to this structure.

How to achieve: In the Issue Navigator, switch to Advanced Searching[28] and run the following query:

```
issue in structure("My todo list")
```

If you want to find issues added to the Default Structure(see page 357), you can omit the structure name:

```
issue in structure()
```

^ up to the list of examples(see page 267)

3.6.3.2 Quick Filter for JIRA Agile's (GreenHopper) Scrum Board to display only low-level issues in a structure

Setup: Suppose that you are using a structure named "Project work breakdown" to organize tasks under higher-level "container" issues that provide an overview of your team's work. In this setting, the actual tasks are at the bottom level of the hierarchy. Also, suppose you are using JIRA Agile's Scrum Board to manage your sprints.

Goal: You want to see only the actual tasks in backlog, hiding the container issues.

How to achieve: Add a Quick Filter[29] to your JIRA Agile (GreenHopper) board with the following JQL:

28 http://confluence.atlassian.com/display/JIRA/Advanced+Searching
29 https://confluence.atlassian.com/display/GH/Configuring+Quick+Filters

```
issue in structure("Project work breakdown", leaf)
```

If your structure is organized such that *two* lower levels matter to you on the JIRA Agile board, you'll search for leaf issues and their parents with this JQL:

```
issue in structure("Project work breakdown", "leaf or parent of leaf")
```

^ up to the list of examples(see page 267)

3.6.3.3 Retrieve all Epics in a certain status and all of their children

Setup: You have a structure named "Enterprise Portfolio" with Epics on the top level, Stories beneath them, and Tasks with their Sub-Tasks occupying the lower levels of the hierarchy.

Goal: You need to see Epics in status *Assigned* with all of their children.

How to achieve: In the Issue Navigator, switch to Advanced Searching[30] and run the following query:

```
issue in structure("Enterprise Portfolio", "issueOrAncestor in [type = Epic and
status = Assigned]")
```

If you want to see these issues in the structure, go to Structure Board(see page 83) and type this query in the Search Area(see page 180) in the JQL mode.

Also, you can type only the last part of the query if you use S-JQL search mode(see page 182):

```
issueOrAncestor in [type = Epic and status = Assigned]
```

^ up to the list of examples(see page 267)

3.6.3.4 Find Test Cases associated with Stories in an active sprint

Setup: Suppose that you have a structure named "Enterprise Portfolio Testing", where you have Epics on the top level, Stories on the second level, then come Test Sub-Tasks, and finally Test Cases.
You are also using JIRA Agile (Greenhopper) to manage your sprints, which contain Stories. The fact that a Test Case is associated with an Story is recorded only in the structure.

Goal: You need to find those Test Cases that are associated with Stories in an active sprint.

How to achieve: You can use Issue Navigator's Advanced Searching[31] capability or open the structure on the Structure Board(see page 83) and use its Search Area(see page 180) in the JQL mode to run this query:

```
issue in structure("Enterprise Portfolio Testing", "[type = 'Test Case'] and ancestor
in [type = Story and sprint in openSprints()]")
```

Or, you can type only the last part of the query if you use S-JQL search mode(see page 182) on the Structure Board:

30 http://confluence.atlassian.com/display/JIRA/Advanced+Searching
31 http://confluence.atlassian.com/display/JIRA/Advanced+Searching

```
[type = 'Test Case'] and ancestor in [type = Story and sprint in openSprints()]
```

^ up to the list of examples(see page 267)

3.6.3.5 Find all issues that are blocking critical issues

Setup: Suppose that you have a structure named "Dependency structure" where parent-child relationship corresponds to dependency: each child blocks its parent. (You might have configured a Links Synchronizer(see page 528) 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[32] capability or open the structure on the Structure Board(see page 83) and use its Search Area(see page 180) in the JQL mode to run this query:

```
issue in structure("Dependency structure", "child of [priority = Critical]")
```

Or, you can type only the last part of the query if you use S-JQL search mode(see page 182) on the Structure Board:

```
child of [priority = Critical]
```

^ up to the list of examples(see page 267)

3.6.3.6 Find all unassigned issues in a part of a project

Setup: Suppose that you use a structure named "Project work breakdown" to break down your project into smaller pieces, so that if you have an issue somewhere in the structure, all of its children at all levels constitute a separate part of a project.

Goal: You are focusing on a part of a project under the issue with key `PROJ-123`, and you want to see unassigned issues in that part of the project.

How to achieve: Use this JQL query to find all unassigned descendants of `PROJ-123`:

```
issue in structure("Project work breakdown", "[assignee is empty] and descendant of
PROJ-123")
```

^ up to the list of examples(see page 267)

3.6.3.7 Top-level view on unfinished parts of a project

Setup: Let's continue with the "Project work breakdown" structure from the previous example. Suppose that there are several top-level issues representing different parts of the project.

Goal: You want to have a view on the parts of the project that are yet unfinished.

---

32 http://confluence.atlassian.com/display/JIRA/Advanced+Searching

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[33] with the following JQL:

```
issue in structure("Project work breakdown", "root and descendants in [resolution is empty]")
```

^ up to the list of examples(see page 267)

3.6.3.8 Find violations of the rule "Tasks must be under Epics or Stories"

Setup: You have a structure named "Planning" where you put issues of types Epic, Story, and Task. Your team follows the convention that Tasks are always put under Epics or Stories. However, as humans are fallible, sometimes a Task ends up being in a wrong place — either on the top level, or under another Task.

Goal: You need to find Tasks that violate the rule, so that you can put them in the right place.

How to achieve: In the Search Area(see page 180) on the Structure Board(see page 83), run the following JQL search(see page 182):

```
issue in structure("Planning", "[type = Task] and parent not in [type in (Epic, Story)]")
```

^ up to the list of examples(see page 267)

3.6.3.9 Find violations of the rule "An issue cannot be resolved if it has unresolved children"

Setup: Suppose that "Planning" is a work breakdown structure. Your team follows the convention that an issue cannot be resolved unless all of its children are resolved.

Goal: You need to find the issues violating this rule.

How to achieve: In the Search Area(see page 180) on the Structure Board(see page 83), run the following S-JQL search(see page 182):

```
[resolution is not empty] and child in [resolution is empty]
```

^ up to the list of examples(see page 267)

3.6.3.10 Find issues that can be resolved because all their children are resolved

Setup: Suppose that "Planning" is a work breakdown structure. Your team follows the convention that once all children of an issue are resolved, the issue can be resolved as well.
The best solution for this would be to use a Status Rollup Synchronizer(see page 534), 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.

---

33 https://confluence.atlassian.com/display/JIRA/Using+Filters

How to achieve: Open the structure on the Structure Board(see page 83). When you paste the query given below into the Search Area(see page 180) (ensure that the JQL mode(see page 182) is selected), the issues that you can resolve will be shown. You can resolve them one by one. Here's the query you need:

```
issue in structure("Planning", "[resolution is empty] and not(child is empty or child
in [resolution is empty])")
```

^ up to the list of examples(see page 267)

3.6.3.11 Get a view of a second (third, ...) level of the hierarchy

Setup: There is a large structure named "Joint Effort" where different users track their issues on several levels: Customer Relations department works with the top-level issues, Project Managers break them down in several issues on the second level, Team Members work with issues under second-level issues.

Goal: Each user wants to see only the relevant part of the structure. Customer Relations department wants to filter out lower-level issues to focus on the top-level ones, and Project Managers sometimes want to focus on just the second-level issues in the context of their parent requests.

How to achieve: use the Search Area(see page 180) on the Structure Board(see page 83) to run the specific queries (ensure that the S-JQL mode(see page 182) is selected.) Toggle the Filter(see page 0) button to hide the issues on the lower levels.

To see top-level issues, run this query:

```
root
```

To see second-level issues (top-level issues will be still displayed, but greyed out), run this query:

```
child of root
```

If you would need to dig even deeper, to see the third level but not the lower ones, you'd use this query:

```
child of (child of root)
```

^ up to the list of examples(see page 267)

3.6.3.12 Get the contents of a folder

Setup: There is a structure with a folder named "Next Release". Issues are placed there manually and then queried via S-JQL for planning purposes (as an Agile board filter, for example).

Goal: The users want to see all issues that are located under the specified folder.

How to achieve: In the Issue Navigator, switch to Advanced Searching[34] and run the following query:

34 http://confluence.atlassian.com/display/JIRA/Advanced+Searching

```
issue in structure("My Structure", "descendant of folder('next release')")
```

Note that the folder name is case-insensitive.

^ up to the list of examples(see page 267)

3.6.4 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[35] capability of JIRA.

List of Structured JQL topics:

3.6.4.1 Multiple instances of items

If there are multiple instances of an item in the structure, some of these instances might match the query, and some might not.

Consider the following structure:

```
TS-239
  TS-42
TS-123
  TS-239
```

Here, issue TS-239 is present two times — one at the root position, and another under another issue. Query `root` wil l match the first instance but not the second one.

This difference is visible when you are filtering in the Structure Widget (see Filter(see page 183)). However, structure() JQL function(see page 293) matches an issue if *at least one* of its instances in the structure matches the S-JQL query. In this example, `issue in structure(root)` will return `TS-239, TS-123.`

3.6.4.2 Constraints

Structure query consists of *constraints*. A constraint matches items in the structure. In the simplest case, the whole structure query consists of a single constraint; for now, we will consider only this case.

There are two types of constraints: *basic* and *relational* constraints.

^ up to the list of S-JQL topics(see page 272)

35 https://confluence.atlassian.com/display/JIRA/Advanced+Searching

3.6.4.3 Basic constraint

A basic constraint matches items that satisfy a condition — regardless of their relative positions to other items.

JQL constraint

JQL constraint matches all issues in the structure that satisfy a JQL query. To specify it, specify the JQL query enclosed in square brackets:

```
[status = Open]
```

`leaf` and `root`

This basic constraint matches items that are located at special positions within the structure.

```
leaf
```

```
root
```

The first constraint matches items at the bottom level of the hierarchy, i.e., items that do not have children (sub-items).
The second constraint matches items at the top level of the hierarchy, i.e., items that do not have a parent.

Specific issue

This kind of basic constraint matches just the referenced issues. If some of the issues are not contained within the structure, they are ignored. If none of the issues are contained within the structure, the constraint matches no issues.

You can specify a comma-separated list of issue keys:

```
TS-129, TS-239
```

One issue key:

```
TS-129
```

Issue ID (or a list of them):

```
19320
```

Function constraint (folder, item)

Functions in S-JQL play the same role as in JQL: it is an extension point, so any vendor can develop their own functions to match items in a custom way.

Structure comes bundled with a few functions: *folder* (matching all folders or folders by name) and *item* (matching all items of the specified type or items by name).

Syntax

A function constraint has a *name* and zero or more *arguments*, depending on the function you are using:

```
folder(Urgent)
```

In the example above, function name is *folder* and its argument is *Urgent.*

You can insert any amount of spaces around the name and arguments:

```
folder ( Urgent )
```

Multiple function arguments should be separated by commas:

```
item(Status, In Progress)
```

If an argument contains commas or parentheses, you need to enclose it in "double quotes" or 'single quotes':

```
item(Status, "Done, Sealed, and Delivered")
folder("NU (non-urgent) issues")
```

The former example matches Status items in structure that are named *Done, Sealed, and Delivered.* If this name wasn't enclosed in quotes, the query would mean that function *item* is given four arguments: *Status*, *Done*, *Sealed* and *and Delivered*.
The latter example matches folders named *NU (non-urgent) issues.* If quotes were not used, the query would be incorrect because the first closing parenthesis would be understood as the end of *folder*'s arguments.

If your argument contains quotes, you need to use another type of quotes to enclose it. Suppose that you need to match a version named *3.0, 3.0.1 "Armageddon"*:

```
item(version, '3.0, 3.0.1 "Armageddon"')
```

You can also escape the quotes using backslash (\). Suppose that the version is named *3.0 Beta 1 "Armageddon's Near":*

```
item(version, '3.0 Beta 1 "Armageddon\'s Near"')
```

If you need to use backslash character on its own, you can escape it with another backslash (\\). Suppose that you need to match a folder named \ *(backslash) and related characters*:

```
folder ('\\ (backslash) and related characters')
```

Note that if you don't need to enclose your argument in quotes, then you don't need to escape quotes or backslashes contained within it:

```
folder (Joe's)
folder ( \ )
```

Finally, if there's only one argument and the argument doesn't contain spaces (or is enclosed in quotes), you can omit the parentheses:

```
folder Urgent
folder "Not urgent"
```

folder()

This function matches folder items in the structure, optionally filtering them by name.

Without arguments, this function matches all folders:

```
folder()
```

With one argument, this function matches folders by name (that you see in the *Summary* column). A folder is matched if its name *starts with* the text specified in the first argument. Difference between capital and small letters is ignored.

For example, the following queries match folders named `My issues`, `Issues for Carol`, and `Non-issues`; and do not match folders named `Is suing` or `Issuance`:

```
folder issue
folder Issue
```

If you specify several words separated by spaces, `folder` will match only folders containing all of these words.

> If you're familiar with how Text Search in structure(see page 180) works, then it's useful to think of this argument in the same way as of the simple query. The only difference is that `folder` doesn't recognize issue keys.

There's an advanced matching option for those who like to use *regular expressions.*

To tell `folder` that you are specifying a regular expression, enclose it in slashes (`/`):

```
folder /i.*ue/
```

If the argument starts with a slash but doesn't end with a slash, regular expression matching doesn't occur, and it's matched as a simple text. If you need to write a simple text search where a text starts and ends with a slash, escape the leading slash with a backslash (\):

```
folder \/???/
```

The query in the example above matches folder `/???/`.

Another advanced topic is how to query for the exact word (e.g., match `issue` but not `issues`).

This is called *strict searching*. Strict searching is turned on when the *search text* starts and ends with a double quote ("). Note, however, that quotes are stripped off from function arguments, since quoting is also used to allow specifying spaces or parentheses in the search text. Thus you'll need to enclose the search text in single quotes (').

```
folder '"issue"'
```

item()

This function matches items of the specified type in the structure, optionally filtering them by name. It is a generalization of `folder()` function to other item types.

The function takes two arguments: *item type* and *name* (optional). The second argument works in the same way as the argument for `folder()` function.

You can reference either standard item types (provided by Structure plugin) or item types provided by third-party plugins.

If you need to match items of all types, use asterisk (*). The following query finds all items that have the word “Infrastructure” in their Summary, regardless of their type:

```
item(*, Infrastructure)
```

Structure provides the following item types:

```
issue
project
version
version-name
project-component
issuetype
status
resolution
priority
label
user
group
date
cf-option
folder
memo
generator
loop-marker
sprint
missing
tempo-account (when Tempo Timesheets plugin is available)
sd-request-type (when Jira Service Management / Jira Service Desk plugin is
available)
```

Structure.Pages[36] 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 is as a fully quailified name. Only if nothing is found, it tries to interpret it as a shortened form.

36 https://wiki.almworks.com/display/pages/Structure.Pages+Extension

> Don't confuse "*matching items of some type*" and "*matching issues that have field value equal to that item*". For example, `item(status, Open)` matches *status Open*, not *issues with status Open*. If you need the latter, use JQL constraint: `[status = Open]`.

Empty constraint

An empty constraint matching no items:

```
empty
```

This constraint plays the same role as JQL's `EMPTY` keyword. It is intended to be used as a sub-constraint(see page 282) in relational constraints, which are discussed further.

^ up to the list of S-JQL topics(see page 272)

3.6.4.4 Negation

Any constraint, basic or relational, can be negated using keyword `NOT`. This produces a constraint that matches all items that the original constraint doesn't:

```
not root
```

matches all items that are not top-level items in the structure.

You can always enclose a constraint in parentheses to ease understanding. So, all items in the structure except issues `TS-129` and `TS-239` are matched by this structure query:

```
not (TS-129, TS-239)
```

^ up to the list of S-JQL topics(see page 272)

3.6.4.5 Relational constraint

A basic constraint matches items that satisfy a condition. A relational constraint matches items *related to* items that satisfy a condition. *Related* corresponds to a relationship between positions of items in the structure, like parent-child.
For example,

```
TS-129
```

is a basic constraint that matches a single issue `TS-129`;

```
child in TS-129
```

is a relational constraint matching items that have `TS-129` as a child (sub-item).

Relational constraint has the form `relation`(see page 279) `operator`(see page 280) `subConstraint`(see page 282). Here, `subConstraint` is a constraint on the relatives of items to be matched; other parts of relational constraint are discussed in the following sections.

> Note that the form of relational constraint is similar to the form of JQL clause, `field operator value`. Indeed, let's describe in English a JQL query `type in (Epic, Story)`: it matches issues having *type* that is *in* values *Epic, Story*.
> Now, let's describe in English a structure query `parent in [type = Epic]`: it matches items having *parent* that is *in* constraint "`type = Epic`".
> As you can see, the form that can be used to describe the structure query is similar to that of JQL.

^ up to the list of S-JQL topics(see page 272)

Relations

S-JQL has the following relations:

- `child`: item is a child (sub-item) of another item in the structure.
- `parent`: item is a parent of another item in the structure.
- `descendant`: item is a descendant (sub- or sub-sub-...-item) of another item in the structure.
- `ancestor`: item is an ancestor (parent, parent-of-parent, or parent-of-parent-...-of-parent) of another item in the structure.
- `sibling`: item is a sibling of another item in the structure. Two items are considered siblings if they are under the same parent item.
- `prevSibling`: item is a previous (preceding) sibling of another item in the structure.
  item *A* is a preceding sibling of item *B* if it is a sibling of *B* and *A* is higher than *B* (*A* comes before *B*.)
- `nextSibling`: item is a next (following) sibling of another item in the structure.
  item *A* is a following sibling of item *B* if it is a sibling of *B* and *A* is lower than *B* (*A* comes after *B*.)
- `self` and `issue` are relations of an item (or an issue) to itself. Their role is explained later, in the `self` and `issue` relation(see page 283) section, because at first one has to learn how operators and sub-constraints work.
  There are also combinations of `issue` and `self` with all other relations, listed for completeness below:

`childOrSelf`	`childOrIssue`
`parentOrSelf`	`parentOrIssue`
`descendantOrSelf`	`descendantOrIssue`
`ancestorOrSelf`	`ancestorOrIssue`
`siblingOrSelf`	`siblingOrIssue`
`prevSiblingOrSelf`	`prevSiblingOrIssue`
`nextSiblingOrSelf`	`nextSiblingOrIssue`

Those familiar with XPath may have recognized these relations; indeed, they work like the corresponding XPath axes.

^ up to the list of S-JQL topics(see page 272)

Operators

These are the operators used in S-JQL:

```
IN, NOT IN, IS, IS NOT, =, !=, OF
```

`operator` specifies how `subConstraint` is applied to `relation`:

1. `IN`, `IS`, and = put constraint on the relatives of a matched item.

   For example, consider

   ```
   child in (TS-129, TS-239)
   ```

   Here, `relation` is `child`, so an item's relative in question is its child in the structure. Thus, an item matches if *at least one of its children is TS-129 or TS-239*.

   There is no difference between these three operators, unlike JQL. Different forms exist to allow for more natural-looking queries with some sub-constraints.

2. `NOT IN`, `IS NOT`, and `!=` are negated versions of `IN`, `IS`, and =. That is, an item is matched if it *is not related to* any item matching `subConstraint`.

   As an important consequence, item that has no relatives is matched.

   For example, consider

   ```
   child not in (TS-129, TS-239)
   ```

   An item matches if *no child is TS-129 nor TS-239*. So, this constraint matches all items that either have no children or do not have any of these two items among their children.

   Using one of these operators in a relational constraint is the same as using `IN` (or `IS`, or =) and negating the whole relational constraint. Thus, the constraint above is equivalent to

   ```
   not (child in (TS-129, TS-239))
   ```

> ⚠ **But**, using one of these operators is **very not** the same as using operator `IN` and negating `subConstraint`!
>
> First, *having relatives other than X* is not the same as *not having relatives X*. Think of it as of relationships in a human family: having a relative other than brother (e.g., a sister) is **not** the same as not having a brother, because one may have both a sister and a brother.
> Second, an item with no relatives is not matched by the transformed query.
>
> For example,
>
> ```
> child in (not (TS-129, TS-239))
> ```
>
> matches all items that have at least one child that is neither `TS-129` nor `TS-239`. That is, the only items that are not matched are leaves and those that have only `TS-129` or `TS-239` as children.

3. `OF` matches the relatives of items that satisfy `subConstraint`.

   For example, consider

   ```
   child of (TS-129, TS-239)
   ```

   An item matches if *it is a child of either `TS-129` or `TS-239`*.

To have a model of how operators `IN` (`IS`, `=`) and `OF` work and to understand the difference between them, consider the table below. Suppose that we take all items in the structure and put each of them, one by one, in column **item**. For each item, we take all of its relatives and put each of them, one by one, in column **relative**. Thus we get pairs of items. We examine each pair, and if one of the components satisfies *subConstraint*, we add the other component to the result set. Which component is added, depends on the operator:

operator	item	relative
`in`	*add to result set*	*satisfies* `subConstraint`
`of`	*satisfies* `subConstraint`	*add to result set*

One may note that for any relation, there is a corresponding "inverse": for example, `child` is the inverse of `parent`, and vice versa. A relational constraint that uses operator `IN` (`IS`, `=`) is equivalent to a relational constraint that uses an inverse relation with operator `OF`. That is,

```
child in (TS-129, TS-239)
```

is the same as

```
parent of (TS-129, TS-239)
```

Again, different forms of expressing the same constraint exist to allow for more natural-looking queries.

^ up to the list of S-JQL topics(see page 272)

Sub-constraints

Any constraint can be used as a sub-constraint, whether basic, relational, or a combination of those(see page 284). For example,

```
child of root
```

selects items on the second level of the hierarchy. To select items on the third level of the hierarchy, you can once again use relation `child` and the previous query as `subConstraint`:

```
child of (child of root)
```

There is a special basic constraint, `empty`, which matches no items. It is used as a sub-constraint to match items that have no relatives as per `relation`.

For example, let's take relation `child` and see what the corresponding relational constraints with different operators mean.

`child is empty`	matches all items that have no children (equivalent of `leaf`)
`child is not empty`	matches all items that have at least one child (equivalent of `not leaf`)
`child of empty`	matches all items that are not children of other items (equivalent of `root`)

Of course, using `leaf` or `root` is more convenient, but you can apply `empty` to any other relation. For instance, `sibling is empty` matches an item if it is the only child of its parent.

^ up to the list of S-JQL topics(see page 272)

`self` and `issues` relations: adding sub-constraint matches to the result set

A relational constraint with relation `self` behaves exactly as its sub-constraint, possibly negated if operator `NOT IN` (`IS NOT`, `!=`) is used.
Thus,

```
self in [status = Open]
```

is equivalent to

```
[status = Open]
```

Similarly,

```
self not in [status = Open]
```

is equivalent to

```
not [status = Open]
```

When combined with another relation, `self` allows to add the items matched by `subConstraint` to the resulting set. For example,

```
descendant of TS-129
```

returns all of the children of `TS-129` at all levels, but does not return `TS-129` itself. To add `TS-129`, use `descendantOrSelf`:

```
descendantOrSelf of TS-129
```

`issue` relation

`issue` is a special case of `self` relation that only matches issues. For instance, if on the top level of the structure you have folders and issues, and you want to hide all folders, you can write this:

```
descendantOrIssue of root
```

This query matches all top-level issues and all their sub-items.

^ up to the list of S-JQL topics(see page 272)

3.6.4.6 Combining constraints with Boolean operators

We can now define a structure query as a *Boolean combination of constraints*, that is, a structure query consists of constraints connected with `AND` and `OR`. When two constraints are connected with `AND`, together they will match issues that are matched by both constraints. This allows you to limit the results. Likewise, when two constraints are connected by `OR`, together they will match issues that are matched by at least one of the constraints. This allows you to expand the results.

Note that `AND` has higher precedence than `OR`. That means that the Structure query

```
leaf or (parent of leaf) and [status = Open]
```

matches all issues that are either leaves, or are parents of leaves in status *Open*. In order to also constrain leaf issues to be in the status *Open*, you need to use parentheses:

```
(leaf or (parent of leaf)) and [status = Open]
```

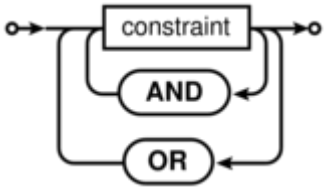
^ up to the list of S-JQL topics(see page 272)

3.6.4.7 Railroad diagrams

As a final piece of reference, here's the S-JQL syntax in the form of railroad diagrams[37].

> ⓘ S-JQL keywords are not case-sensitive, and all underscores in keywords are optional.

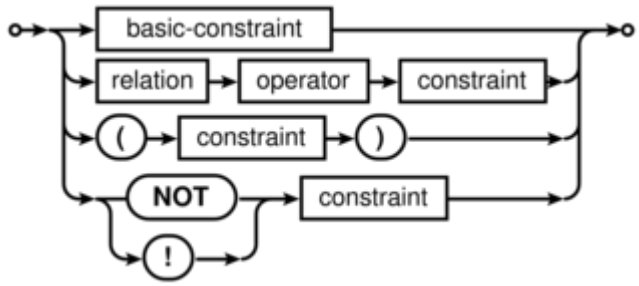
structure-query



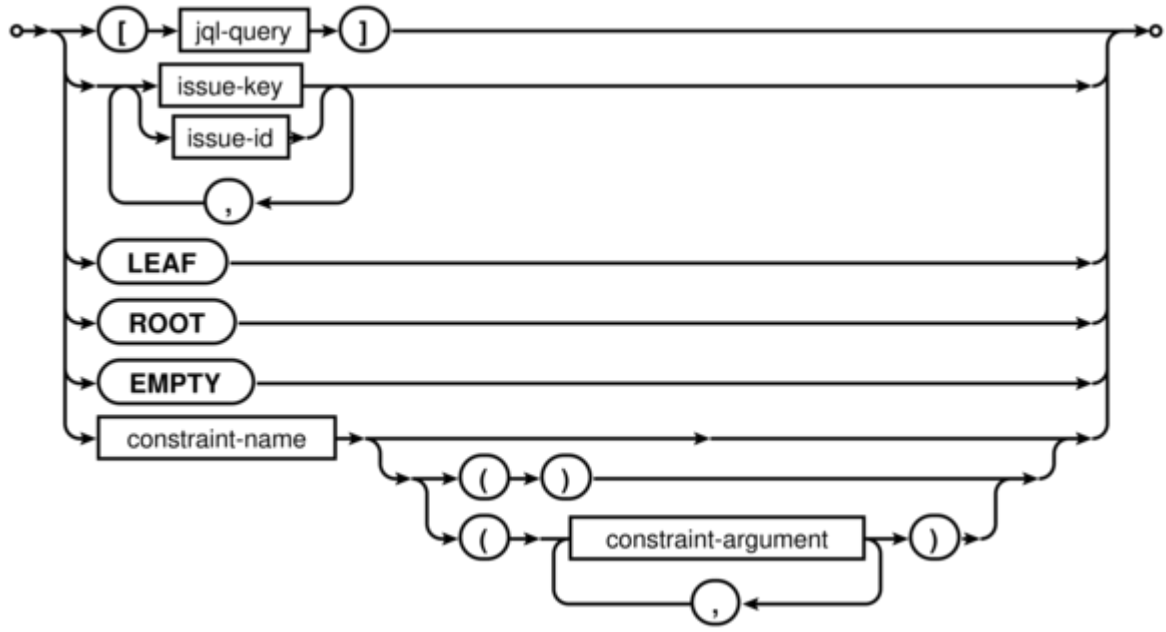
> ⓘ S-JQL admits using && and & in place of `AND`, as well as || and | in place of `OR`.

37 http://en.wikipedia.org/wiki/Railroad_diagram

constraint



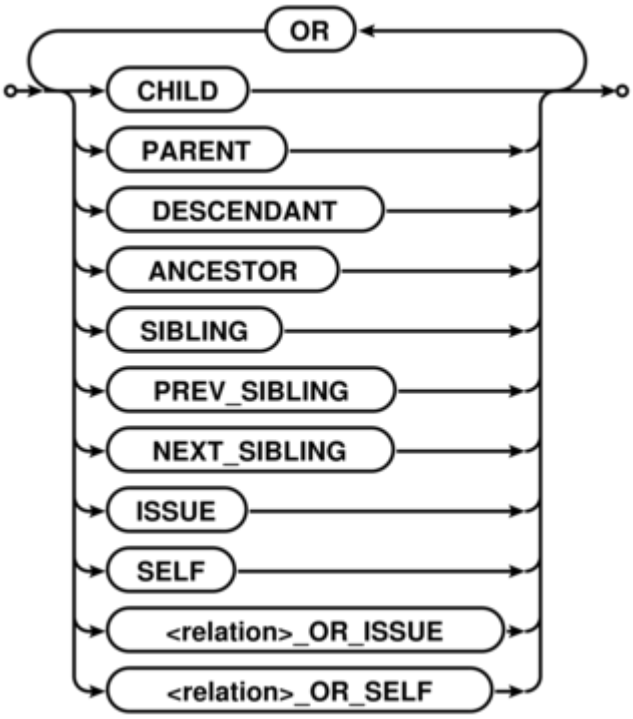
basic-constraint



- `jql-query` is any valid JQL query.
- `issue-key` is any valid JIRA issue key.
- `issue-id` is any valid JIRA issue ID.
- `constraint-name` is the name of the function constraint: either bundled with Structure (`folder`, `item`, or `row_id`) or provided by a Structure extension (plugin).
- `constraint-argument` is one of the following:
  - either a sequence of non-whitespace characters
  - or quoted text (inside "double quotes" or 'single quotes'), where quotes can be escaped via backslash: `\"`, `\'`; backslash itself can be escaped: `\\`.

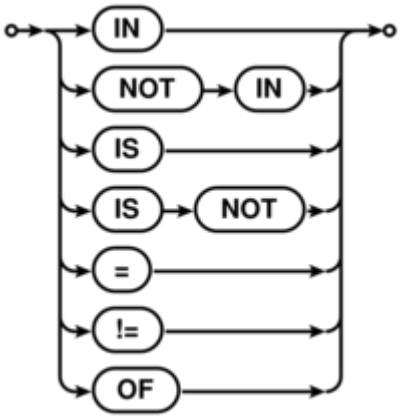
  See also Function constraint - Syntax(see page 0).

relation



S-JQL admits using || and | in place of OR.

operator



^ up to the list of S-JQL topics(see page 272)

3.6.4.8 List of S-JQL keywords

In this article, all S-JQL keywords are listed in all of their spelling variants. This is intended for developers creating their own S-JQL function, because function name must not coincide with the existing keyword.

```
!
!=
&
&&
(
)
,
=
[
]
ancestor
ancestor_or_issue
ancestor_or_issues
ancestor_or_self
ancestorOrIssue
ancestorOrIssues
ancestorOrSelf
ancestors
ancestors_or_issue
ancestors_or_issues
ancestors_or_self
ancestorsOrIssue
ancestorsOrIssues
ancestorsOrSelf
and
child
child_or_issue
child_or_issues
child_or_self
childOrIssue
childOrIssues
childOrSelf
children
children_or_issue
children_or_issues
children_or_self
childrenOrIssue
childrenOrIssues
childrenOrSelf
descendant
descendant_or_issue
descendant_or_issues
descendant_or_self
descendantOrIssue
descendantOrIssues
descendantOrSelf
descendants
descendants_or_issue
descendants_or_issues
descendants_or_self
descendantsOrIssue
```

```
descendantsOrIssues
descendantsOrSelf
empty
following_sibling
following_sibling_or_issue
following_sibling_or_issues
following_sibling_or_self
following_siblings
following_siblings_or_issue
following_siblings_or_issues
following_siblings_or_self
followingSibling
followingSiblingOrIssue
followingSiblingOrIssues
followingSiblingOrSelf
followingSiblings
followingSiblingsOrIssue
followingSiblingsOrIssues
followingSiblingsOrSelf
in
is
issue
issue_or_ancestor
issue_or_ancestors
issue_or_child
issue_or_children
issue_or_descendant
issue_or_descendants
issue_or_following_sibling
issue_or_following_siblings
issue_or_next_sibling
issue_or_next_siblings
issue_or_parent
issue_or_parents
issue_or_preceding_sibling
issue_or_preceding_siblings
issue_or_prev_sibling
issue_or_prev_siblings
issue_or_previous_sibling
issue_or_previous_siblings
issue_or_sibling
issue_or_siblings
issue_or_sub_issue
issue_or_sub_issues
issueOrAncestor
issueOrAncestors
issueOrChild
issueOrChildren
issueOrDescendant
issueOrDescendants
issueOrFollowingSibling
issueOrFollowingSiblings
```

```
issueOrNextSibling
issueOrNextSiblings
issueOrParent
issueOrParents
issueOrPrecedingSibling
issueOrPrecedingSiblings
issueOrPreviousSibling
issueOrPreviousSiblings
issueOrPrevSibling
issueOrPrevSiblings
issueOrSibling
issueOrSiblings
issueOrSubIssue
issueOrSubIssues
issues
issues_or_ancestor
issues_or_ancestors
issues_or_child
issues_or_children
issues_or_descendant
issues_or_descendants
issues_or_following_sibling
issues_or_following_siblings
issues_or_next_sibling
issues_or_next_siblings
issues_or_parent
issues_or_parents
issues_or_preceding_sibling
issues_or_preceding_siblings
issues_or_prev_sibling
issues_or_prev_siblings
issues_or_previous_sibling
issues_or_previous_siblings
issues_or_sibling
issues_or_siblings
issues_or_sub_issue
issues_or_sub_issues
issuesOrAncestor
issuesOrAncestors
issuesOrChild
issuesOrChildren
issuesOrDescendant
issuesOrDescendants
issuesOrFollowingSibling
issuesOrFollowingSiblings
issuesOrNextSibling
issuesOrNextSiblings
issuesOrParent
issuesOrParents
issuesOrPrecedingSibling
issuesOrPrecedingSiblings
issuesOrPreviousSibling
```

```
issuesOrPreviousSiblings
issuesOrPrevSibling
issuesOrPrevSiblings
issuesOrSibling
issuesOrSiblings
issuesOrSubIssue
issuesOrSubIssues
leaf
leaves
next_sibling
next_sibling_or_issue
next_sibling_or_issues
next_sibling_or_self
next_siblings
next_siblings_or_issue
next_siblings_or_issues
next_siblings_or_self
nextSibling
nextSiblingOrIssue
nextSiblingOrIssues
nextSiblingOrSelf
nextSiblings
nextSiblingsOrIssue
nextSiblingsOrIssues
nextSiblingsOrSelf
not
null
of
or
parent
parent_or_issue
parent_or_issues
parent_or_self
parentOrIssue
parentOrIssues
parentOrSelf
parents
parents_or_issue
parents_or_issues
parents_or_self
parentsOrIssue
parentsOrIssues
parentsOrSelf
preceding_sibling
preceding_sibling_or_issue
preceding_sibling_or_issues
preceding_sibling_or_self
preceding_siblings
preceding_siblings_or_issue
preceding_siblings_or_issues
preceding_siblings_or_self
precedingSibling
```

```
precedingSiblingOrIssue
precedingSiblingOrIssues
precedingSiblingOrSelf
precedingSiblings
precedingSiblingsOrIssue
precedingSiblingsOrIssues
precedingSiblingsOrSelf
prev_sibling
prev_sibling_or_issue
prev_sibling_or_issues
prev_sibling_or_self
prev_siblings
prev_siblings_or_issue
prev_siblings_or_issues
prev_siblings_or_self
previous_sibling
previous_sibling_or_issue
previous_sibling_or_issues
previous_sibling_or_self
previous_siblings
previous_siblings_or_issue
previous_siblings_or_issues
previous_siblings_or_self
previousSibling
previousSiblingOrIssue
previousSiblingOrIssues
previousSiblingOrSelf
previousSiblings
previousSiblingsOrIssue
previousSiblingsOrIssues
previousSiblingsOrSelf
prevSibling
prevSiblingOrIssue
prevSiblingOrIssues
prevSiblingOrSelf
prevSiblings
prevSiblingsOrIssue
prevSiblingsOrIssues
prevSiblingsOrSelf
root
roots
self
self_or_ancestor
self_or_ancestors
self_or_child
self_or_children
self_or_descendant
self_or_descendants
self_or_following_sibling
self_or_following_siblings
self_or_next_sibling
self_or_next_siblings
```

```
self_or_parent
self_or_parents
self_or_preceding_sibling
self_or_preceding_siblings
self_or_prev_sibling
self_or_prev_siblings
self_or_previous_sibling
self_or_previous_siblings
self_or_sibling
self_or_siblings
self_or_sub_issue
self_or_sub_issues
selfOrAncestor
selfOrAncestors
selfOrChild
selfOrChildren
selfOrDescendant
selfOrDescendants
selfOrFollowingSibling
selfOrFollowingSiblings
selfOrNextSibling
selfOrNextSiblings
selfOrParent
selfOrParents
selfOrPrecedingSibling
selfOrPrecedingSiblings
selfOrPreviousSibling
selfOrPreviousSiblings
selfOrPrevSibling
selfOrPrevSiblings
selfOrSibling
selfOrSiblings
selfOrSubIssue
selfOrSubIssues
sibling
sibling_or_issue
sibling_or_issues
sibling_or_self
siblingOrIssue
siblingOrIssues
siblingOrSelf
siblings
siblings_or_issue
siblings_or_issues
siblings_or_self
siblingsOrIssue
siblingsOrIssues
siblingsOrSelf
sub_issue
sub_issue_or_issue
sub_issue_or_issues
sub_issue_or_self
```

```
sub_issues
sub_issues_or_issue
sub_issues_or_issues
sub_issues_or_self
subIssue
subIssueOrIssue
subIssueOrIssues
subIssueOrSelf
subIssues
subIssuesOrIssue
subIssuesOrIssues
subIssuesOrSelf
|
||
```

3.6.5 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[38] and Advanced Searching Functions[39].

If a user does not have access to structure(see page 478), they will not be able to create new queries with the `structure()` function and existing queries will have `structure()` function return an empty set. However, the user will still see `structure()` function offered in the JQL completion drop-down.

To specify a structure condition in JQL, use the following format:

issue in structure(*structureName*$_{opt}$, *structureQuery*$_{opt}$**)**

Function arguments:

structureName	*Optional*	The name of the structure. If you omit the structure name, system-wide Default Structure(see page 357) will be searched.
structureQuery	*Optional*	Use this parameter to select only a part of the structure. This parameter specifies a *Structure Query* in a language similar to JQL, Structured JQL(see page 266).

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.

38 https://confluence.atlassian.com/display/JIRA/Advanced+Searching
39 https://confluence.atlassian.com/display/JIRA/Advanced+Searching+Functions

3.6.5.1 Function arguments need to be quoted if they contain spaces or non-letters

As dictated by the syntax of JQL, you'll need to enclose structure name or structure query in 'single quotes' or "double quotes" if they contain spaces or non-letters.

What if *structure name* or *structure query* itself contains quotes?

If structure name or structure query contains quotes of one kind, you need to enclose them with a different kind of quotes. That is, if structure query contains double quote, you'll need to enclose it in single quotes. Alternatively, you can escape quote with a backslash: `\"`.

Example 1

Suppose you need to find all issues that are directly under issues in status *Awaiting Deployment.*

In plain JQL, issues in this status can be found via this query: `Status = "Awaiting Deployment"`. Note that since status name contains spaces, JQL requires us to enclose it in quotes.

According to S-JQL Reference(see page 272), the corresponding Structure query would be `child of [Status = "Awaiting Deployment"]`.

That means that you need to enclose this Structure query with single quotes:

```
issue in structure("My personal structure", 'child of [Status = "Awaiting
Deployment"]')
```

Note that the following will **not** work:

```
issue in structure("My personal structure", "child of [Status = "Awaiting
Deployment"]")
```

Example 2: escaping with backslash

In the following example, the query returns issues that are directly under issues assigned to fix version named *3.0 "Armageddon"*.

```
issue in structure("My personal structure", "child of [fixVersion = '3.0
\"Armageddon\"']")
```

3.6.5.2 Backward compatibility with `structure()` JQL function prior to Structure 2.4

Prior to Structure 2.4, `structure()` JQL function did not take structure query as an argument; you could specify only one issue key or ID, and you would get the referenced issue along with all of its children at all levels. As you might have noticed, this old-style usage can be interpreted as a structure query, but according to the rules of S-JQL, it would return just the referenced issue without its children. To maintain backward compatibility, any structure

query in Structure 2.4 that consists of a single basic constraint that references issues by their keys or IDs matches not only these issues, but all of their children as well.

That means that if you were using JQL of the form

```
issue in structure("My personal structure", TS-129)
```

then in Structure 2.4 this query will still return `TS-129` and all of its children at all levels (provided that `TS-129` is added to the structure.)

If this backward compatibility bites you (if, say, you need to check whether an issue is added to a structure), prepend the structure query with `issue in`:

```
issue in structure("My personal structure", "issue in TS-129")
```

This JQL will match only `TS-129` if it is in the structure.

3.7 Columns and Views

A **view** defines which **columns** are displayed in Structure and in what configuration.

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

https://www.youtube.com/watch?v=WXTEPs-hPyU

The following sections will show you how to customize your columns and views to provide exactly the information you need.

3.7.1 Structure Columns

Structure provides a number of columns that display information about issues in the structure. You can customize(see page 331) the displayed columns by adding new columns, changing each column configuration, or switching to a new view(see page 337).

Out of the box, Structure provides the following columns:

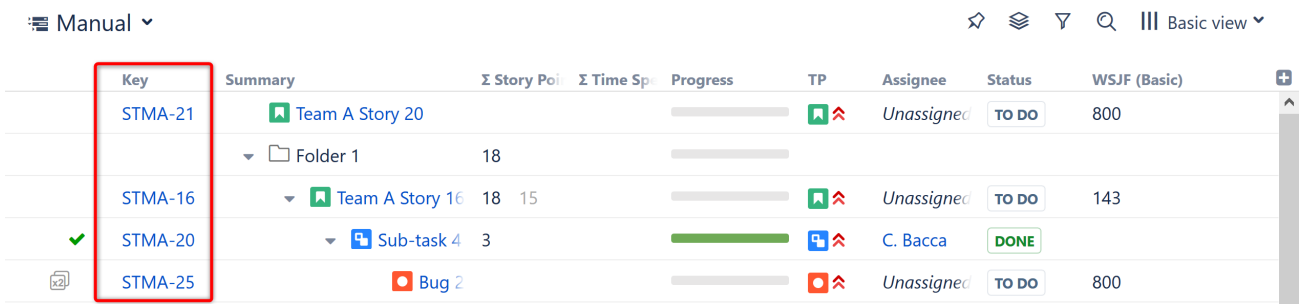
- Issue Key Column(see page 296)
- Summary Column(see page 296)
- Field Columns(see page 297)
- Icons Column(see page 299)
- Progress Column(see page 300)
    - Progress Based on Time Tracking(see page 301)
    - Progress Based on Resolution Only(see page 305)
    - Progress Based on Status(see page 307)
    - Progress Based on Percent Field(see page 310)

- Images Column(see page 314)
- Work Logged Column(see page 315)
- Flags Column(see page 317)
- Jira Actions Column(see page 317)
- Sequential Index Column(see page 318)
- Notes Column(see page 319)
- Totals Columns(see page 320)
- Time in Status Column(see page 321)
- Status Category Column(see page 325)
- Count Sub-Items Column(see page 326)
- Count Leaves Column(see page 327)
- Transition Dates(see page 327)
- Query Match Column(see page 328)
- Last Comment Column(see page 330)

Structure also contains extension API(see page 842), so the selection of available columns may be extended by a third-party plugin.

3.7.1.1 Issue Key Column

The Issue Key column displays the issue key for issues. For other types of items it remains empty.



Compact View

If the project key is large, the issue key column may get too wide. You can configure the Issue Key column to replace the project key with a small avatar icon of the project.

To enable compact view for the Issue Key column, open the column options(see page 331) and select **Compact View.**

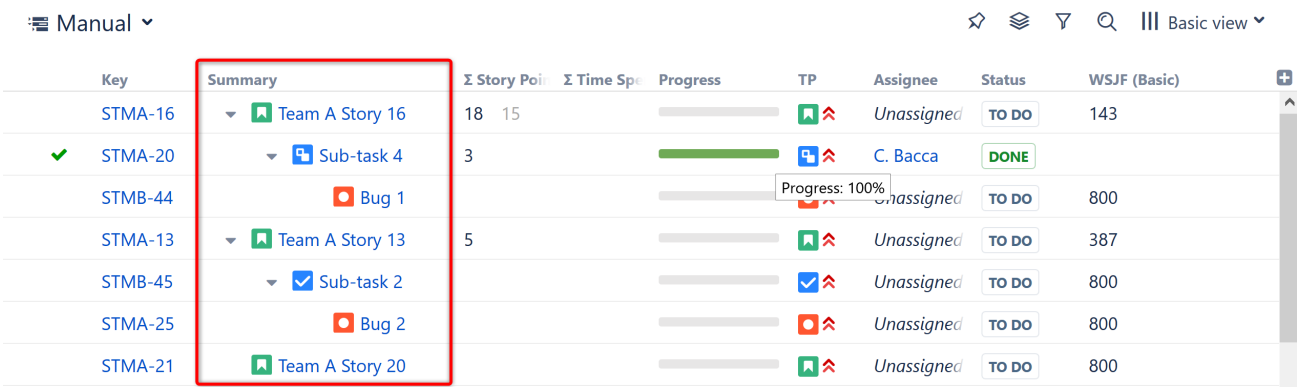
3.7.1.2 Summary Column

The Summary column displays the issue summary and, optionally, part of the issue description. For folders, it shows the folder name.

Summary can be edited right in the structure widget(see page 114) and it is the only field required for creating new issues(see page 110).

Structure Hierarchy

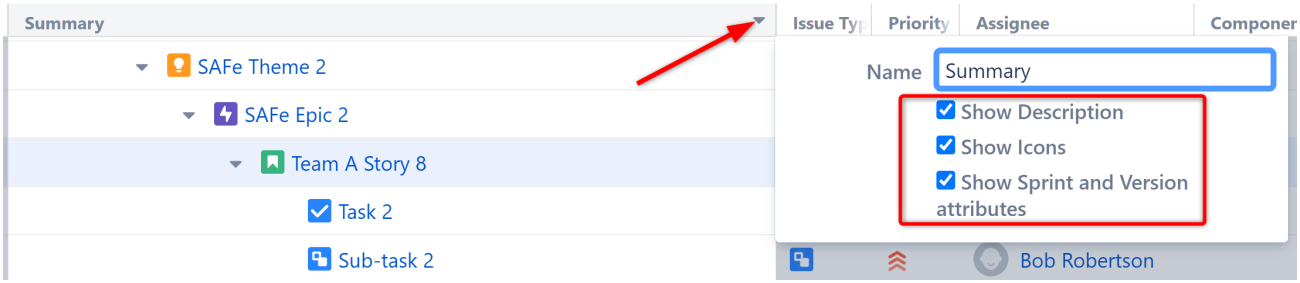
The Summary column is also where the structure hierarchy is displayed. The text for sub-items is indented relative to their parent items.



Customizing the Summary Column

To customize the Summary column, click the arrow icon in the column header and select which items should be displayed:

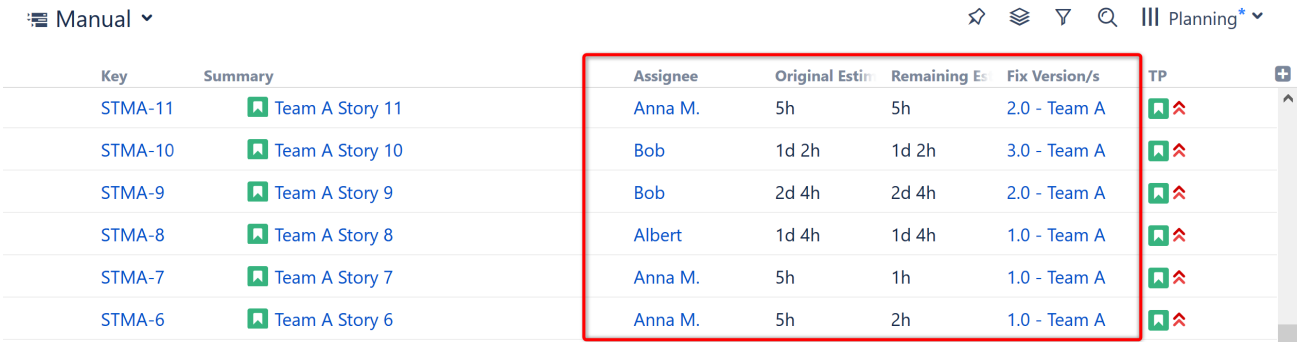
- Item Description
- Item Icons
- Sprint and Version attributes (Sprint start date, end date, and sprint goal; Fix Version start date, release date, and description)



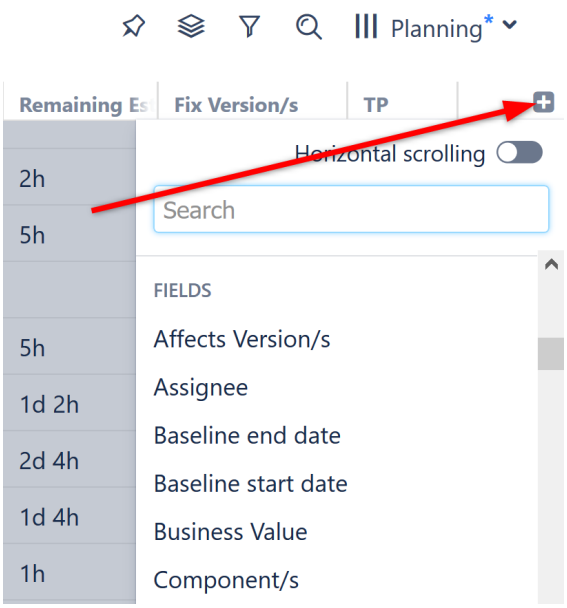
> The Summary column cannot be removed from the Structure grid or reconfigured to a different column type, because it displays the hierarchy.

3.7.1.3 Field Columns

For each issue field in your Jira, Structure offers a column that displays that field's value.

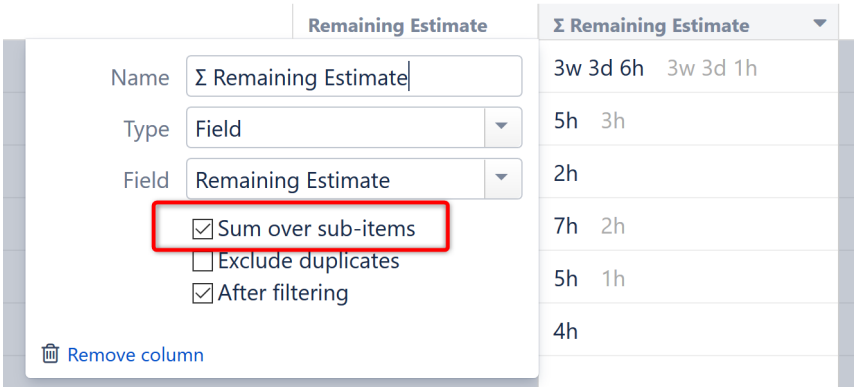


Each Structure view(see page 336) contains a different set of field columns, with a focus on a specific task or business need. But each team has different needs - if you don't see the issue fields you're looking for, you can simply add new columns(see page 331) to include them in your structure.



Displaying Aggregate Values (Totals)

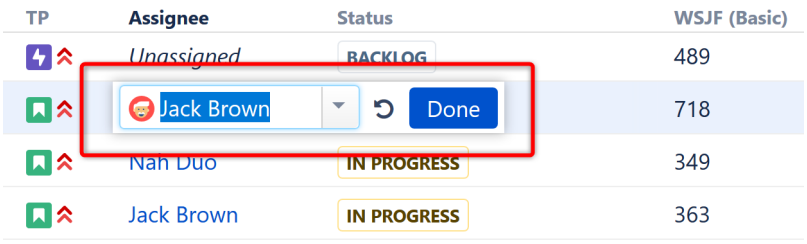
To display aggregate totals(see page 320) for a numeric or time-tracking field, check the **Sum over sub-items** box.



If the Sum over sub-issues option is unavailable for a given field, then the aggregate cannot be calculated.

Editing Field Values

Most issue fields can be edited directly in the Structure panel – you can edit a field's value(see page 114) by double-clicking it (if the field is added to the Edit Screen in Jira).

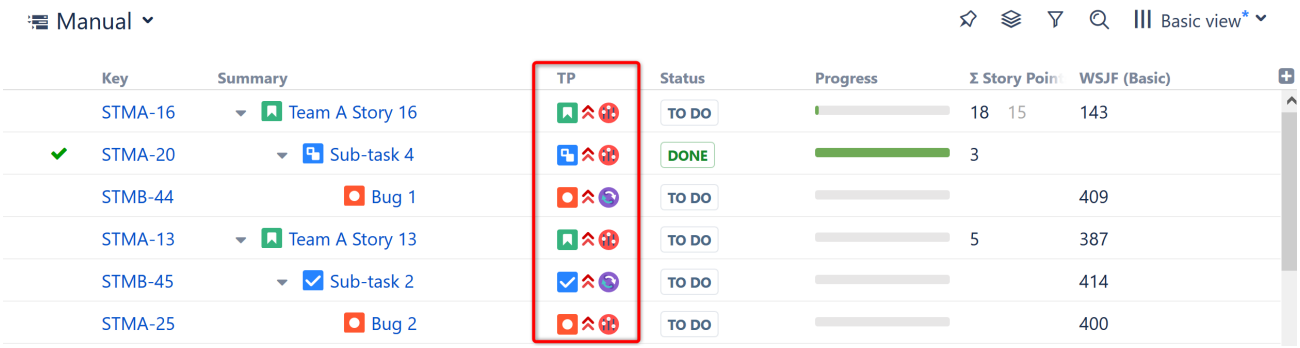


When an aggregate value is displayed, double-clicking the field will allow you to edit the issue's own value.

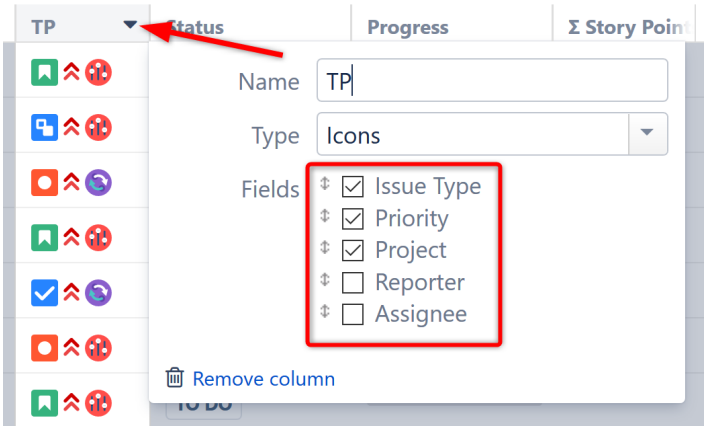
> ⚠ When editing Status, you can only select statuses that are allowed according to the workflow and that have no required fields or dialogs to show in the corresponding transition.

3.7.1.4 Icons Column

The Icons column displays icons for issue type, priority, project, reporter and assignee.

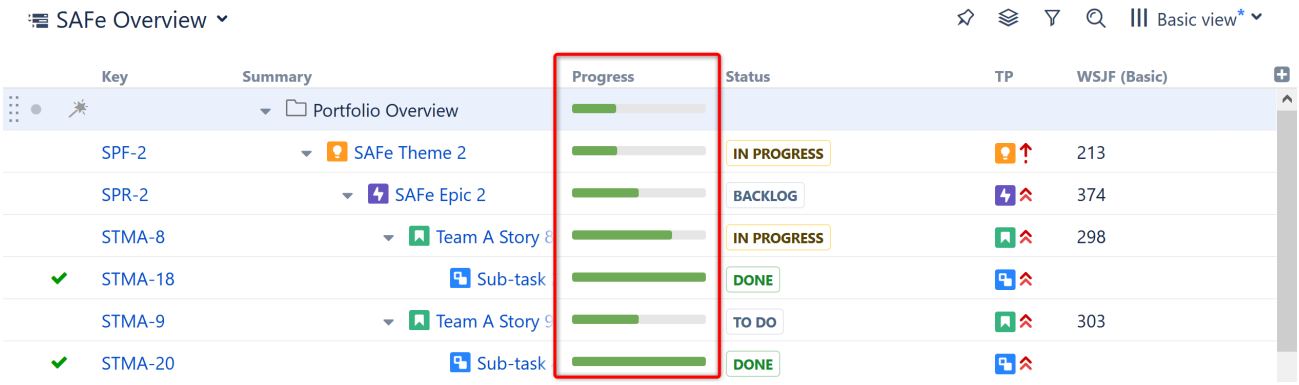


You can choose which icons are displayed and what order they appear in. Simply click the downward-facing arrow next to the column header and select which fields to include. To rearrange your icons, drag and drop the items in the Fields list.



3.7.1.5 Progress Column

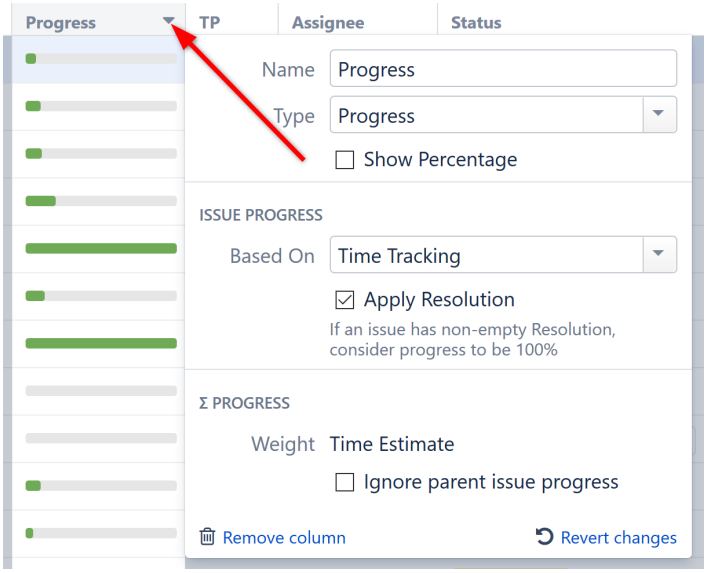
The Progress column displays an aggregate issue progress, which is calculated based on values from the issue and its sub-issues.



Progress can be based on:

- Time tracking,(see page 301)
- Resolution,(see page 305)
- Status,(see page 307) or
- A custom percentage field(see page 310).

You can select which type of progress column to include using the Add Column(see page 331) menu. Once a progress column has been added to your structure, you can change how it's calculated or customize its configuration using the column configuration panel(see page 332).



Progress is a custom Structure column, not available in the Issue Navigator or other standard Jira views.

How is Progress Calculated?

Progress calculations can be customized in two parts:

1. How the individual issue progress is calculated, regardless of its position in the structure.
2. How the progress of sub-issues are aggregated and combined with the individual progress of the parent issue.

Individual Issue Progress Calculation

There are several progress calculation modes. You can select which one to use in the **Based On** option:

- Progress Based on Time Tracking(see page 301) — The progress is calculated based on the issue's Resolution field, time tracking data and the progress of its sub-issues.
- Progress Based on Resolution Only(see page 305) — The progress is calculated based on the issue's Resolution field and the progress of its sub-issues.
- Progress Based on Status(see page 307) — The progress is determined by the issue's Status field. Custom percentage values can be assigned to each status.
- Progress Based on Percent Field(see page 310) — The progress is assigned to each issue manually in a custom field, and progress is aggregated for parent issues.

Total Progress Calculation

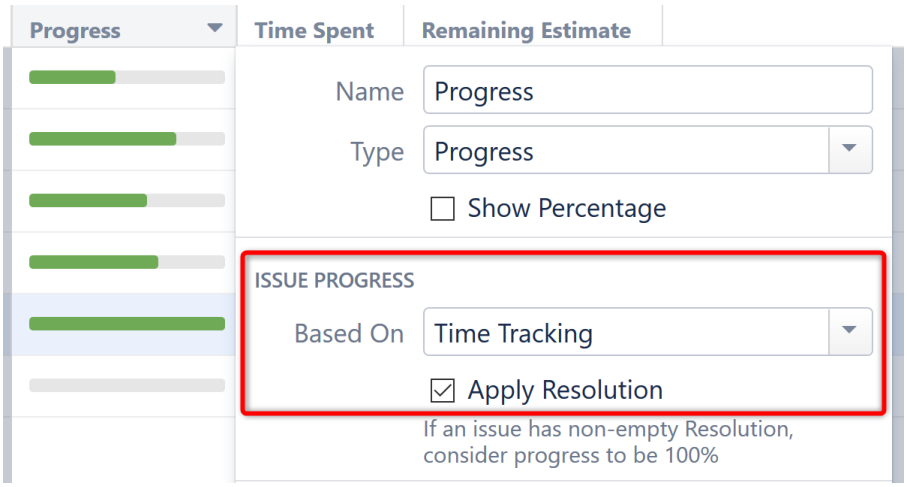
When individual issue progress is calculated based on Status(see page 307), Percent Field(see page 310), or Resolution Only(see page 305), you can specify how sub-issue progresses are aggregated into their parent issue progress. This is defined by the **Weight** option:

- **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.



Calculating Progress for Issue Without Sub-Issues

If the issue does not have sub-issues:

- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100%.
- Otherwise, if the issue has time tracking information, the progress is calculated proportionally to this issue completion%: `(Time Spent) / (Time Spent + Remaining Estimate)`
- Otherwise, the progress is 0%.

Calculating Progress for Issue with Sub-Issues

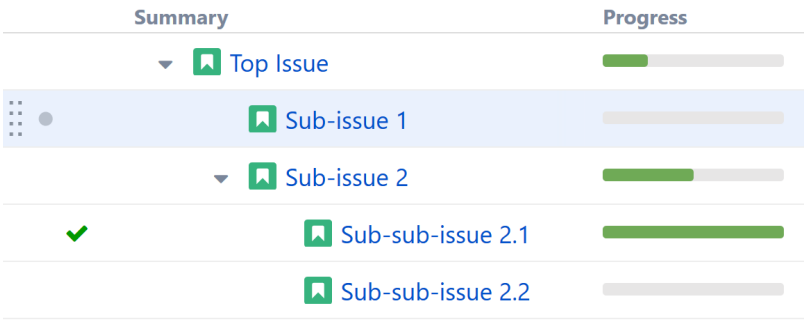
If the issue has sub-issues:

- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100% - regardless of the sub-issues' progress.
- If the issue and its sub-issues do not have estimates or work logged (or if time tracking is turned off), the progress is calculated as the average of the sub-issues' progresses.
- If time tracking is used and all issues have an estimate (either original estimate or remaining estimate), the estimates and total work logged are summed up and the progress is calculated as the total completion %: `(Total Time Spent) / (Total Time Spent + Total Remaining Estimate)`
    - If a sub-issue does not have time tracking information, it is counted in as an average sub-issue, based on the mean total time (time spent + remaining estimate) of its siblings.

If the issue has both its own time tracking information and sub-issues with progress, and if **Ignore Parent Issue Progress** is turned off, the issue's own progress value is counted as if it was the progress of another sub-issue.

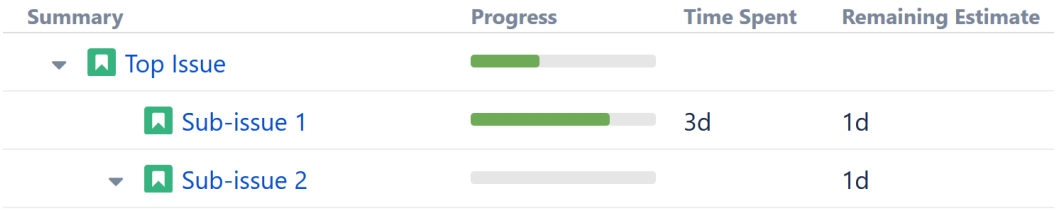
Examples

1. Without Time Estimates



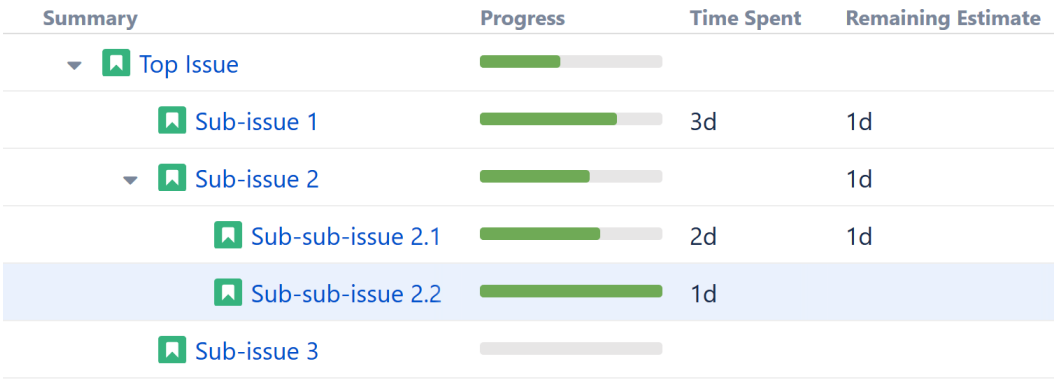
Issue	Explanation	Progress
Sub-sub-issue 2.1	The issue is resolved (indicated by the green mark), so it is complete.	100%
Sub-issue 2	It has two sub-issues with 100% and 0% progress; the total progress is the average of the two.	50%
Top issue	It has two Sub-issues: sub-issue 1 is 0% done and Sub-issue 2 is 50% done; the mean value is 25%.	25%

2. With Time Tracking Information



Issue	Explanation	Progress
Sub-issue 1	It has 3 days of work logged with 1 day remaining, so its progress is `time spent / total time = 3 / (3 + 1)`.	75%
Sub-issue 2	This issue does not have any work logged, is not resolved and does not have sub-issues.	0%
Top issue	The top issue has a total time spent of 3 days (work logged on Sub-issue 1) and 2 total days remaining (estimates on Sub-issue 1 and Sub-issue 2), so its progress `3 / (3 + 2)`.	60%

3. More Complex Example

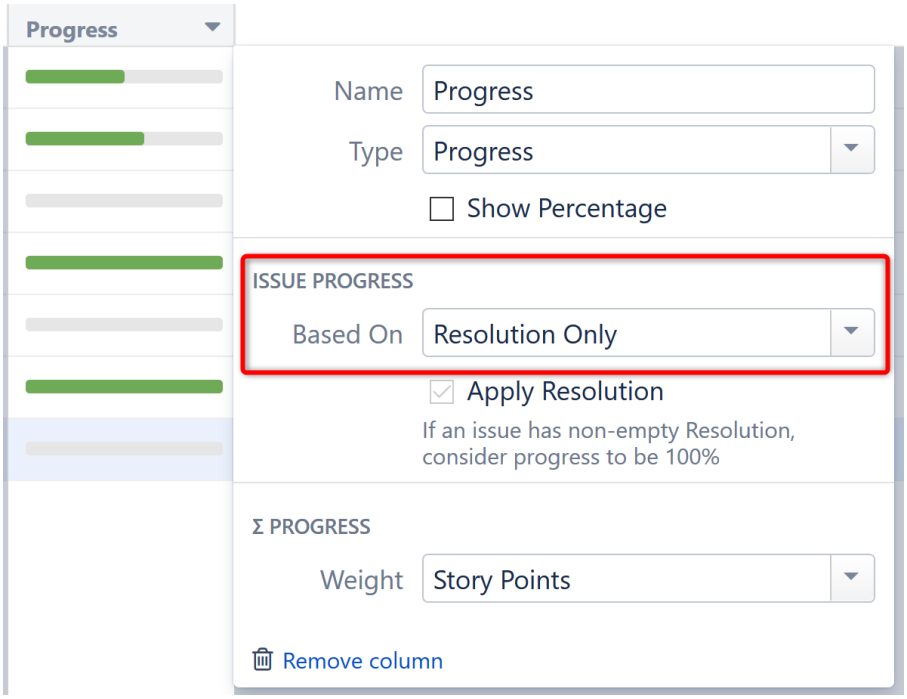


Issue	Explanation	Progress
Sub-sub-issue 2.1	It has 2 days of work logged with 1 day remaining, so its progress is `2 / (2 + 1)`.	66%
Sub-sub-issue 2.2	This issue has 1 day of work logged and no work remaining - so even though it is not resolved, it's considered completed.	100%
Sub-issue 2	It has total time spent of 3 days, and total remaining estimate of 2 days (the remaining time from Sub-sub-issue 2.1 and its own 1 day, which is considered additional work). The progress is `3 / (3 + 2)`.	60%
Sub-issue 1	This one has 3 days of work logged and 1 day remaining, so its progress is `3 / (3 + 1)`.	75%

Issue	Explanation	Progress
Top issue	The obvious total time spent is 6 days with a total remaining estimate of 3 days (the count from all sub-issues on all levels). But there's also *Sub-issue 3*, which does not have any estimates or work logged, so it gets estimated based on the average of its siblings - *Sub-issue 1* and *Sub-issue 2*. The progress of the top issue is calculated as follows: • The average between the total time of *Sub-issue 1* (3 + 1 = 4 days) and the total time of *Sub-issue 2* (3 + 2 = 5 days) is 4.5 days. So *sub-issue 3* is treated as if it has a total time of 4.5 days. • Since Sub-Issue 3 has 0% progress (because there is no time logged), it is also treated as if it has a remaining estimate of 4.5 days. • Top Issue is then calculated as having a total time spent of 6 days and total remaining time of 7.5 days, so its progress is calculated as `6 / (6 + 7.5)`.	44%

Progress Based on Resolution Only

When Issue Progress is based on Resolution Only within a Progress column, the progress is calculated based on the issue's Resolution field and the progress of its sub-issues.



Calculating Progress for Issue Without Sub-Issues

If the issue does not have sub-issues:

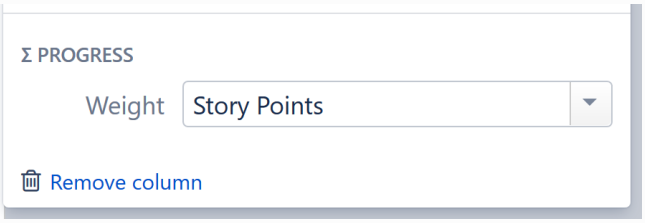
- If the issue's Resolution field is not empty, the progress is 100%.
- Otherwise, the progress is 0%.

Calculating Progress for Issue with Sub-Issues

If the issue does have sub-issues:

- If the issue's Resolution field is not empty, the progress is 100% - regardless of the sub-issues' progress.
- Otherwise, the issue's progress is the weighted average of its sub-issues.

You can specify how an issue's weight is determined in the column configuration panel(see page 332).



Example

Resolution Only with Story Points
When Issue Progress is based on Resolution Only and the Weight is determined by Story Points:

- Individual progress is 0% or 100% based on Resolution field
- Total progress is calculated as a weighted average, with weights contained in the *Story Points* field

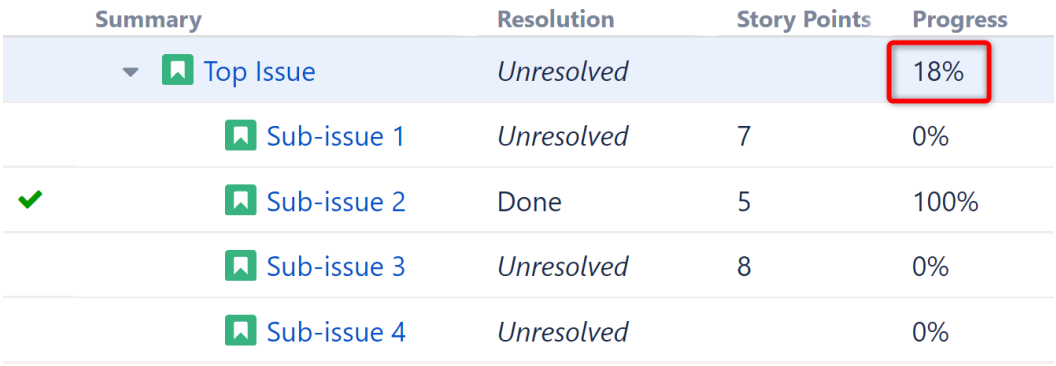
Summary	Resolution	Story Points	Progress
Top Issue	*Unresolved*		
Sub-issue 1	*Unresolved*		
Sub-sub-issue 2.1	*Unresolved*	2	
✔ Sub-sub-issue 2.2	Fixed	3	
Sub-issue 2	*Unresolved*	1	

Issue	Explanation	Progress
Sub-sub-issue 2.2	This issue is resolved (indicated by the green mark) - so it is complete.	100%

Issue	Explanation	Progress
Sub-issue 1	It has two sub-issues with 0% and 100% progress, and story points are 2 and 3 respectively. So the total progress is the weighted average value of (0 x 2 + 100 x 3) / (2 + 3).	60%
Top issue	It has two sub-issues: Sub-issue 1 (60% done) and Sub-issue 2 (0% done), and their cumulative story points are (2 + 3) and 1, respectively. So the progress is (60 x 5 + 0 x 1) / (5 + 1).	50%

Estimating Total Progress When Weight Values Are Missing

When calculating a parent's total progress, children without weight values are calculated using an average weight of their siblings.



Summary	Resolution	Story Points	Progress
Top Issue	*Unresolved*		18%
Sub-issue 1	*Unresolved*	7	0%
✔ Sub-issue 2	Done	5	100%
Sub-issue 3	*Unresolved*	8	0%
Sub-issue 4	*Unresolved*		0%

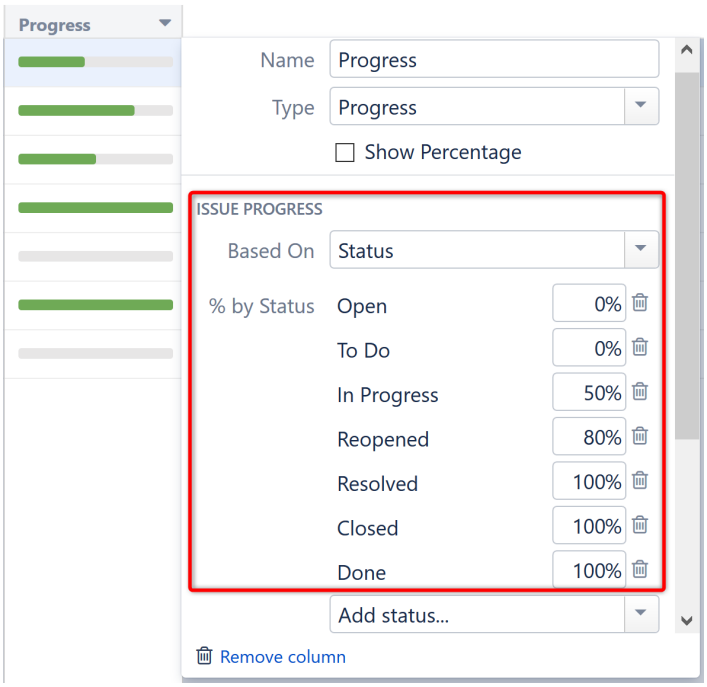
In the above example, we might assume that the progress of Top Issue would be 25%, because 5 story points are completed, out of total 20 (7+5+8) - but that only works if Sub-issue 4 requires no work to complete. That's probably not the case!

In order to account for the work Sub-issue 4 will require, Structure assumes an average number of story points (or whatever the selected weight value), based on the values of its siblings. In this case, that's the average of 7, 5, and 8 - or 6.66.

When Structure calculates the progress for Top Issue, it compares the completed story points of all siblings (5) against the combined actual and estimated story points (7+5+8+6.66), for a total progress of 18%.

Progress Based on Status

When Issue Progress is based on Status within a Progress column, the progress is determined by the issue's Status field. Custom percentage values can be assigned to each status.



Calculating Progress for Issue Without Sub-Issues

If the issue does not have sub-issues:

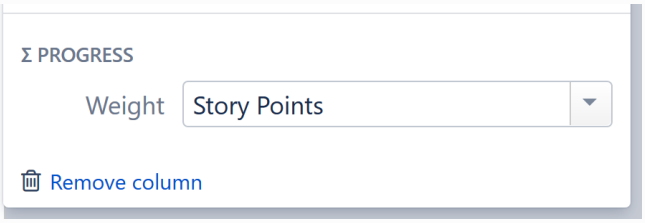
- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100%.
- If the issue's Status is assigned a value (%) in the column configuration, the progress is equal to that value.
- Otherwise, the progress is undefined. The issue neither shows any progress, nor affects the progress of its parent issue.

Calculating Progress for Issue with Sub-Issues

If the issue does have sub-issues:

- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100% - regardless of the sub-issues' progress.
- Otherwise, the issue's progress is the weighted average of its sub-issues.
    - If the issue has both its own status and sub-issues with progress, and if **Ignore Parent Issue Progress** is turned off, the issue's own progress value is counted as if was the progress of another sub-issue.

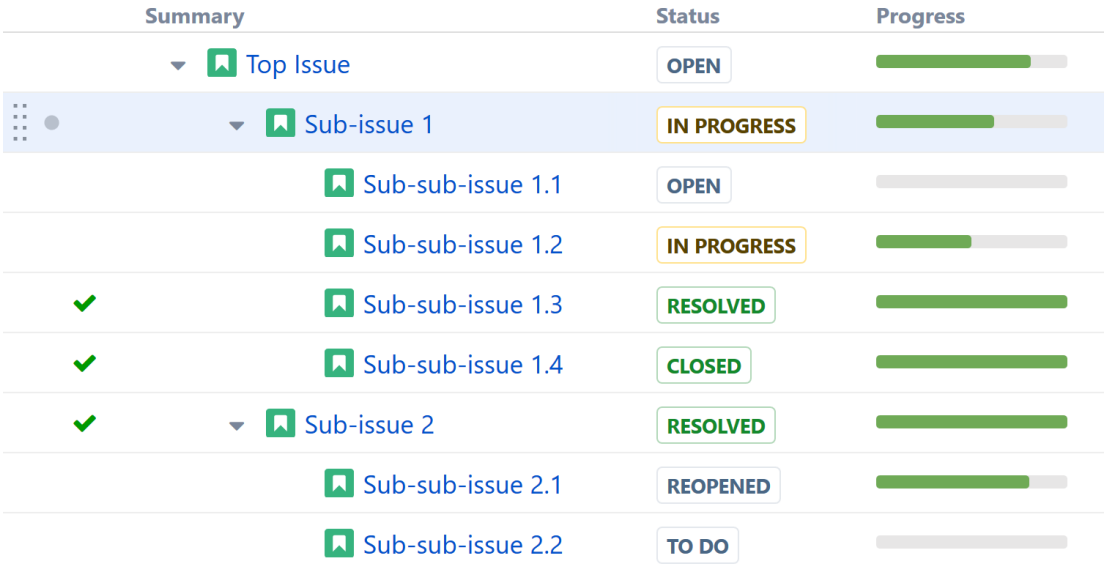
You can specify how an issue's weight is determined in the column configuration panel(see page 332).



⚠ If some of the statuses do not have any percentage configured, the issue progress is considered undefined. Issues with those statuses will neither show any progress, nor affect the progress of their parent issues.

Example

Progress Based on Status, All Sub-Issues Are Equal
In this example, statuses have the following percentages: Open = 0%, In Progress = 50%, Resolved or Closed = 100%, Reopened = 80%. **Apply Resolution** is turned on, and **Ignore Parent Issue Progress** is turned on.

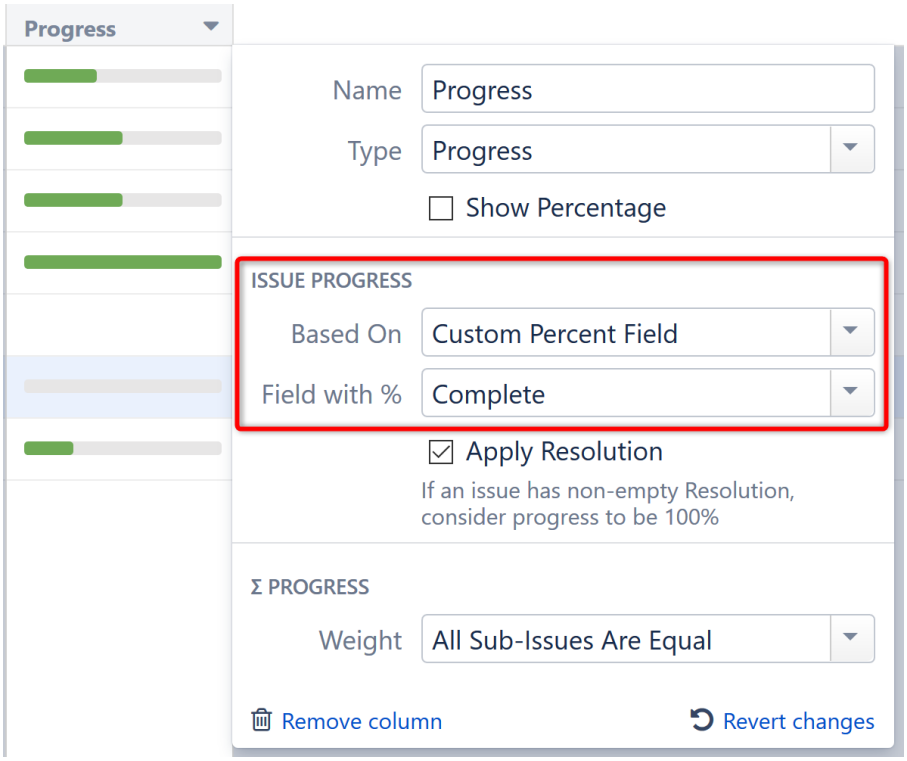


Issue	Explanation	Progress
Sub-sub-issue 1.1	This issue is Open, so the progress is 0%.	0%
Sub-sub-issue 1.2	This issue is In Progress, so progress is 50%.	50%
Sub-sub-issue 1.3	This issue is Resolved, so progress is 100%. Also, according to the workflow, it has a non-empty Resolution, which also means it's complete.	100%
Sub-sub-issue 1.4	This issue is Closed, so progress is 100%. Also, according to workflow, it has a non-empty Resolution, which also means it's complete.	100%

Issue	Explanation	Progress
Sub-issue 1	The average progress of its sub-issues is (0+50+100+100)/4. The issue's own status is In Progress, but it's percentage is ignored because of the "Ignore parent issue progress in aggregation" option.	63%
Sub-sub-issue 2.1	This issue is Reopened, so progress is 80%.	80%
Sub-sub-issue 2.2	This issue is Open, so progress is 0%.	0%
Sub-issue 2	The average progress of its sub-issues is (80+0)/2 = 40%. But the issue itself has a Resolution and the "Issues with Resolution are 100% done" option is turned on, so this overrides the sub-issues' progress and makes the issue complete.	100%
Top issue	It has two sub-issues: Sub-issue 1 is 63% done and Sub-issue 2 is 100% done. Average progress is (63+100)/2.	82%

Progress Based on Percent Field

When Issue Progress is based on a custom percent field, the progress is assigned to each issue manually in a custom field, and progress is aggregated for parent issues.



You can use any numeric Jira custom field to store the current progress % – a value from 0 to 100.

Calculating Progress for Issue Without Sub-Issues

If the issue does not have sub-issues:

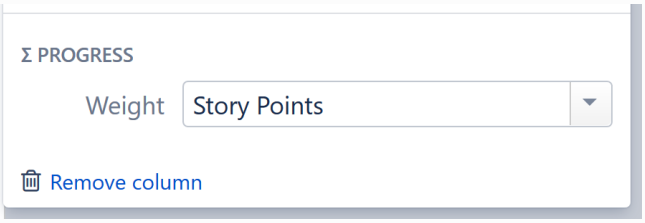
- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100%.
- If the issue's Custom Field value is not empty and is between 0 and 100, it's considered as the completion progress in percents.
- If the issue's Custom Field value is less than 0, the progress is 0%; if greater than 100, the progress is 100%.
- Otherwise, the progress is undefined. The issue neither shows any progress, nor affects the progress of its parent issue.

Calculating Progress for Issue with Sub-Issues

If the issue does have sub-issues:

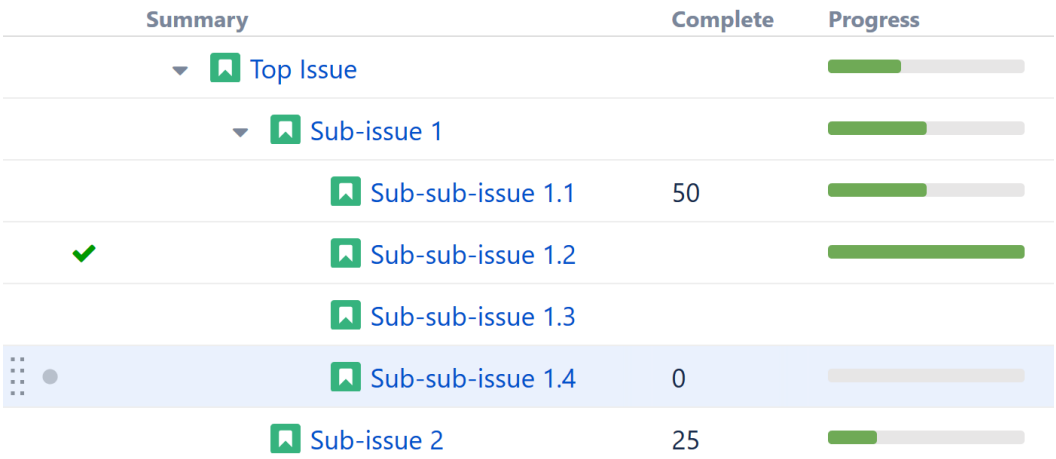
- If the issue's Resolution field is not empty, and **Apply Resolution** is turned on, the progress is 100% – regardless of the sub-issues' progress.
- If the issue's Custom Field value is not empty, it's considered as that issue's completion progress in percents (from 0 to 100) – regardless of the sub-issues' progress.
- Otherwise, the issue's progress is the weighted average of its sub-issues.

You can specify how an issue's weight is determined in the column configuration panel(see page 332).



Examples

1. Percent Field, All Sub-Issues Are Equal
With a Custom field named *Complete*, total progress based on **All Sub-Issues Are Equal**, and **Apply Resolution** turned on.



Issue	Explanation	Progress
Sub-sub-issue 1.1	This issue is 50% complete as specified by the custom field.	50%
Sub-sub-issue 1.2	This issue is resolved (indicated by the green mark) - so it is complete, even if the "Complete" field is empty.	100%
Sub-sub-issue 1.3	This issue has no progress information (neither "Resolution" nor "Complete" fields), so progress is undefined and not counted at all.	n/a

Issue	Explanation	Progress
Sub-sub-issue 1.4	This issue has a 0 "Complete" value, which means it's 0% complete.	0%
Sub-issue 1	It has four sub-issues, but 1.3 is ignored. So the total progress is the average of the rest: (50 + 100 + 0) / 3.	50%
Sub-issue 2	The issue is 25% complete as specified by the custom field.	25%
Top issue	It has two sub-issues: Sub-issue 1 is 50% done and Sub-issue 2 is 25% done. So the progress is the average of the two: (25 + 50) / 2.	38%

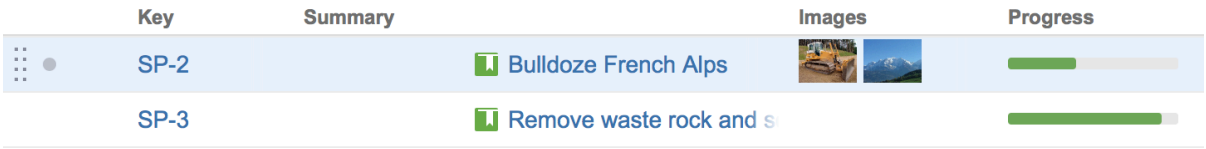
2. Percent Field, Story Points
With a Custom field named *Complete*, total progress based on the field *Story Points*, and **Apply Resolution** turned on.

Issue	Explanation	Progress
Sub-sub-issue 1.1	This issue is 50% complete as specified by the custom field.	50%
Sub-sub-issue 1.2	This issue is resolved (indicated by the green mark) - so it is complete, even if the "Complete" field is empty.	100%

Issue	Explanation	Progress
Sub-sub-issue 1.3	This issue has no progress information (neither "Resolution" nor "Complete" fields), so progress is undefined and not counted at all.	n/a
Sub-sub-issue 1.4	This issue has a 0 "Complete" value, which means it's 0% complete.	0%
Sub-issue 1	It has four sub-issues, and weight is based on story points: • 1.1 has 2 story points • 1.2 has 3 story points • 1.3 is ignored (because its progress is undefined) • 1.4 has no story points, so it's weight is calculated as the mean of its siblings (2 and 3 = 2.5) The total progress is the weighted average of 1.1, 1.2 and 1.4: (50 x 2 + 100 x 3 + 0 x 2.5) / (2 + 3 + 2.5).	53%
Sub-issue 2	The issue is 25% complete as specified by the custom field.	25%
Top issue	It has two sub-issues: Sub-issue 1 is 53% done and has 7.5 story points; Sub-issue 2 is 25% done and has 1 story point. So the progress is calculated as (53 x 7.5 + 25 x 1) / (7.5 + 1).	50%

3.7.1.6 Images Column

The Images column displays small thumbnails of attached image files and allows users to view those images in a pop-up dialog.



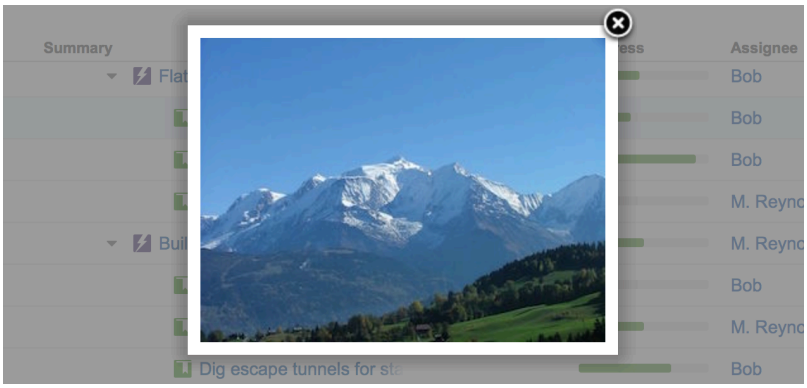
Viewing Full-size Images

Using your mouse:

1. Click the image thumbnail to see the full-size image in a dialog box.
2. If more than one image file is attached, use the left and right arrows to move between them.
3. Click the close button at the top right corner to close the full-size image view.

Using your keyboard:

1. Select the issue that contains images.
2. Press **i,i** ("i" twice) to view the first image.
3. Press → to go to the next image, ← to go to the previous image.
4. Press **Esc** to close the full-size image view.



Images from Wikipedia

3.7.1.7 Work Logged Column

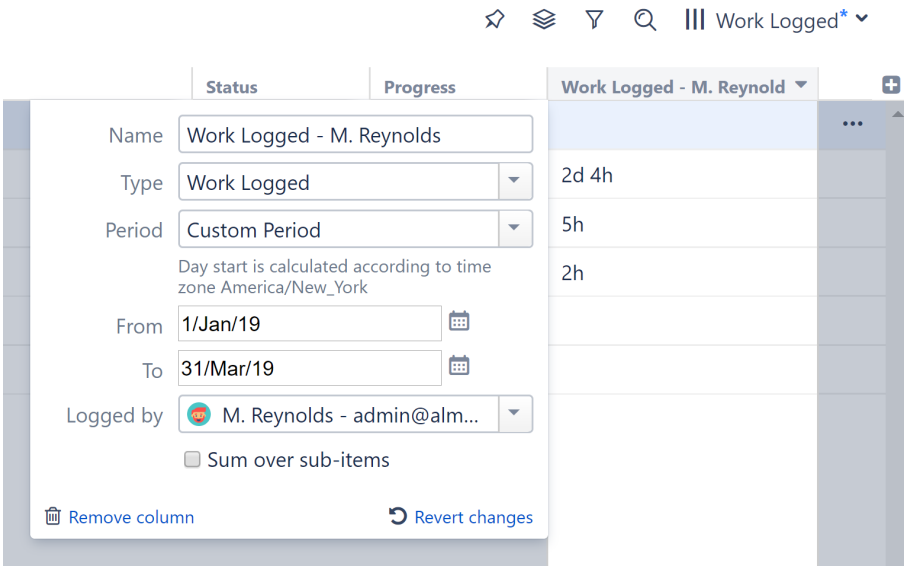
The Work Logged column displays the sum of time spent on an issue over a specific period of time and, optionally, by a specific user, group or project role.

Summary	Work Today
Epic 1	
Story 1	4h
Story 2	5h
Bug 1	2h
Story 3	1d

Customizing a Work Logged Column

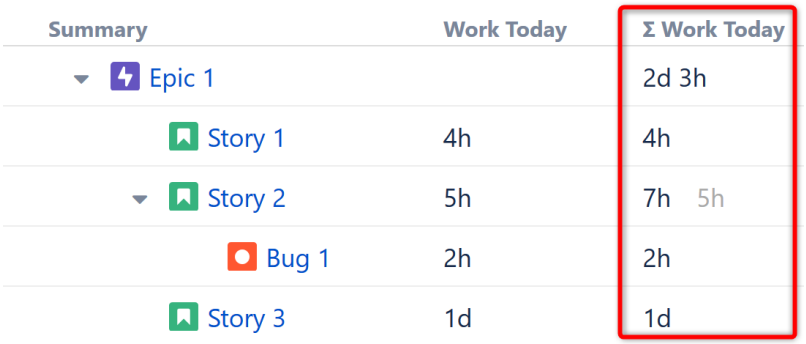
Each Work Logged column can be customized to display precisely the information you need to see. Using the column configuration panel[40], you can select from one of the predefined time periods or specify a custom period. Additionally, you can choose to narrow the results to a particular user, group or project role.

40 https://wiki.almworks.com/display/structuremaster/Customizing+Columns#CustomizingColumns-config-panel



Displaying Aggregate Values

The Work Logged column can also display aggregated values, calculated as the sum of time spent over sub-issues.



To display aggregated values:

- From the Add Columns menu, select **Σ Worked Logged...**, or
- Starting from an existing Work Logged column, open the column configuration panel[41] 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[42].

- To view worked logged based on your own time zone, create your own instance of the Work Logged column.

41 https://wiki.almworks.com/display/structuremaster/Customizing+Columns#CustomizingColumns-config-panel
42 https://confluence.atlassian.com/display/JIRA/Managing+your+User+Profile

3.7.1.8 Flags Column

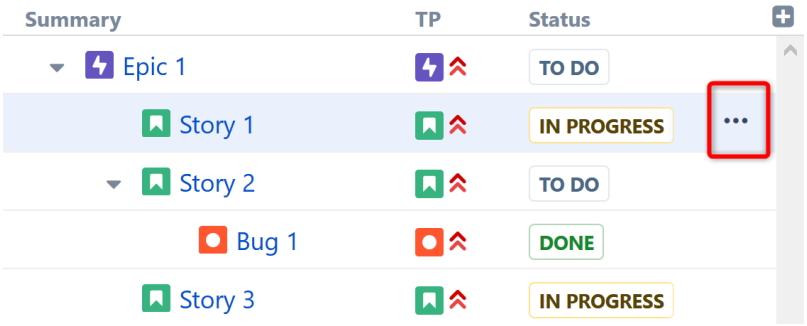
Flags are the small icons displayed at the left side of issue rows to mark specific issue states.

Structure displays the following flags:

✔	**Resolved** flag means that the issue is in a Done status category. Such issues are considered completed and filtered out by the Unresolved Quick Transformation(see page 188).
🔒	**Read-only** flag means that the current user does not have permission to add, remove or rearrange sub-items of the flagged issue. This flag is only displayed when: • The structure is configured to require Edit Issue permission on Parent Issue(see page 0), • The user does not have permission to edit this issue(see page 114).

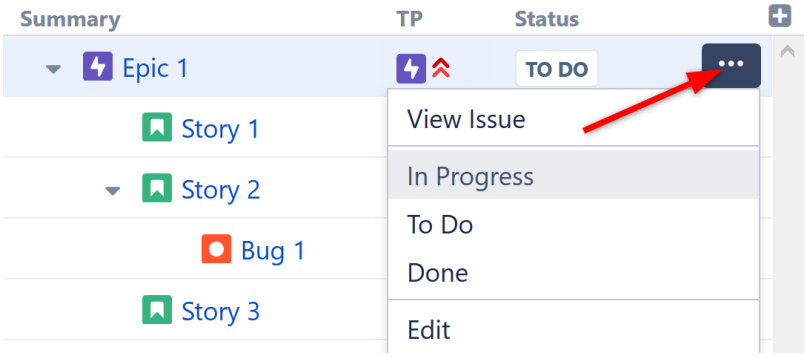
3.7.1.9 Jira Actions Column

The Jira Actions column allows you to open a menu with available Jira actions for the issue. This column works like the similar column on the Jira's Issue Navigator page and lets you log work, apply workflow actions and use other Jira actions(see page 121) available for the issue.



To use the Jira Actions menu:

- Click the **...** button and select the desired action with the mouse, or
- Use the keyboard shortcut **Alt+Down Arrow** to open the menu for the currently selected issue and then use the **Up** and **Down** arrow keys and **Enter** key to select the action



⚠ The **...** button is only visible when the row is selected or when you hover over the Jira Actions column with the mouse.

3.7.1.10 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			TO DO	800	1
STMB-31	Story 1			IN PROGRESS	326	1.1
STMB-32	Story 2			TO DO	296	1.2
STMB-43	Bug 1			DONE		1.2.1
STMA-35	Bug 2			TO DO	800	1.2.2
STMB-41	Epic 2			TO DO	144	2
STMB-37	Story 4			IN PROGRESS	326	2.1
STMA-36	Task			TO DO	800	2.1.1
STMB-39	Story 6			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:

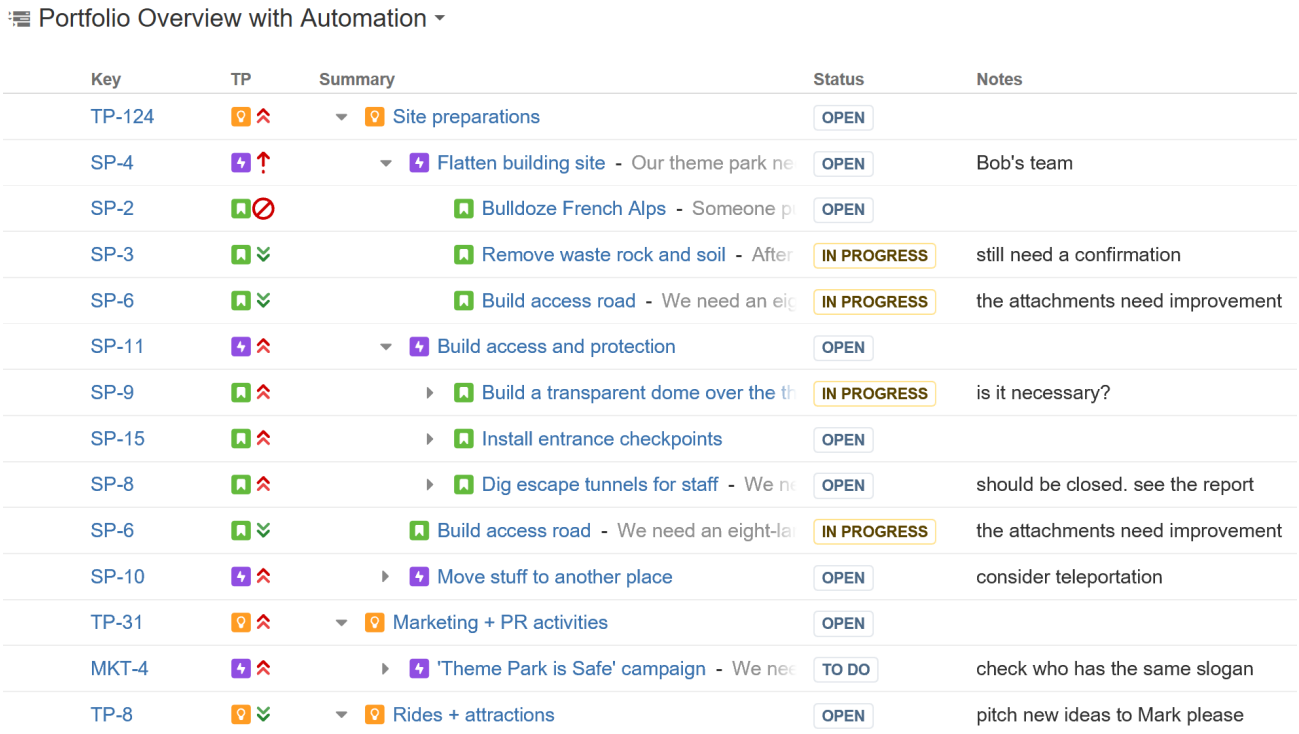
- **Epic 1** has a sequential index of 1, because it is the first item in the top level of the hierarchy.
- **Story 2** has a sequential index of 1.2, because it is the second child under **Epic 1**.
- **Bug 1** has a sequential index of 1.2.1, because it is the first child under **Story 2**.
- **Task** 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(see page 183) – 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(see page 130) will change the sequential index, because they change the underlying structure.

3.7.1.11 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.



The values in the Notes columns are **per-structure, per-item**. This means that:

- Text entered as Notes for some issue in one structure will not be seen for that issue in another structure. You may have different notes for the same issue in different structures.
- If an item occurs several times in a structure, they will have the same value in the Notes column.

The Notes column is great for leaving issue-level notes. For project-level notes, consider adding a Memo item(see page 97) to the structure.

Permissions

Data stored in the Notes column is considered to be a property of the selected structure. That has the following effect on the permissions.

Who can view notes?

To be able to see the notes, the user needs to have:

- View access to the structure that stores the notes.
- View access to the item (issue, project, etc.)

Who can edit notes?

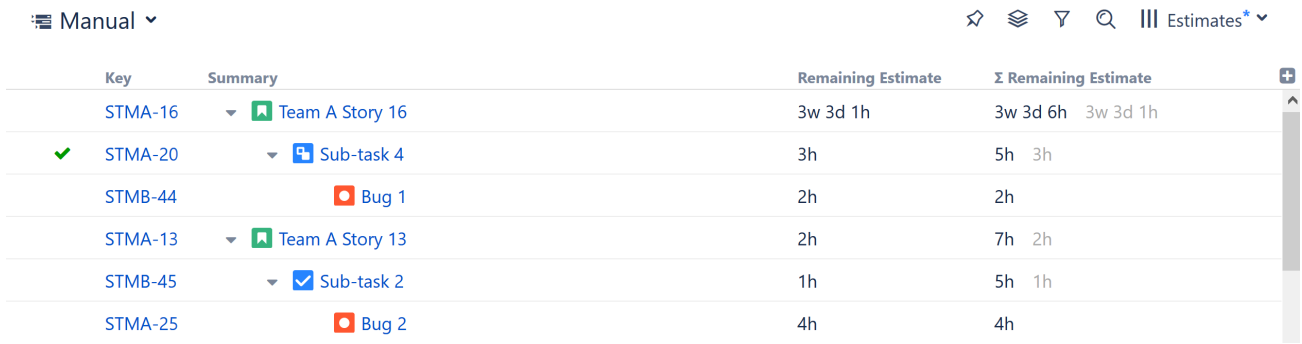
To be able to edit the notes, the user needs to have:

- Edit access to the structure that stores the notes.
- View access to the item (issue, project, etc.)

A user might have permission to edit notes, even if he or she does not have permission to edit the issue. By creating their own structure, a user can leave notes for issues they can't edit.

3.7.1.12 Totals Columns

Totals columns provide aggregate values for all numeric and time-tracking fields. These are calculated as the sum of the current item's field value and those of its sub-issues.

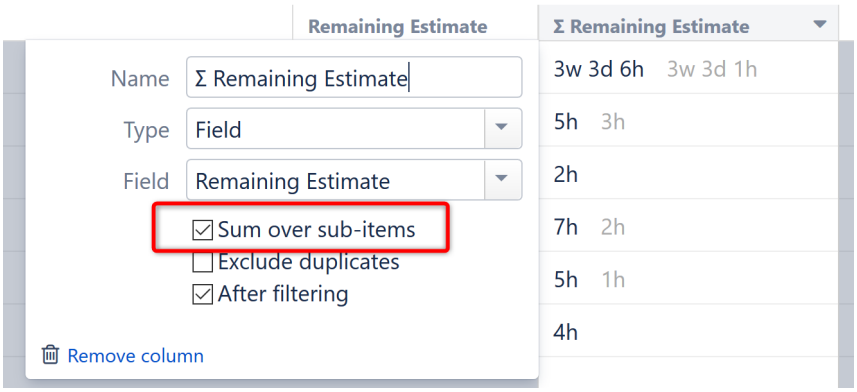


When using an aggregate column:

- When an aggregate value is displayed for an issue that also has its own value in the field, its own value is displayed next to the aggregate value in a gray color.
- Since issues can be present multiple times in a structure, you can select whether you want to count every instance of an issue or count it just once. By default, duplicates are counted each time they appear. To exclude them, select **Exclude Duplicates** in the column configuration panel(see page 0).

Sum over sub-items

All aggregate columns have the Sum over sub-items box checked.



This is what makes them aggregate columns. Unchecking this box will change the column from an aggregate column to it's corresponding Field column.

Build Your Own Aggregate Columns

You can also build your own aggregate columns for most numeric and time-tracking fields. Simply open the column configuration panel(see page 0) and select **Sum over sub-issues**.

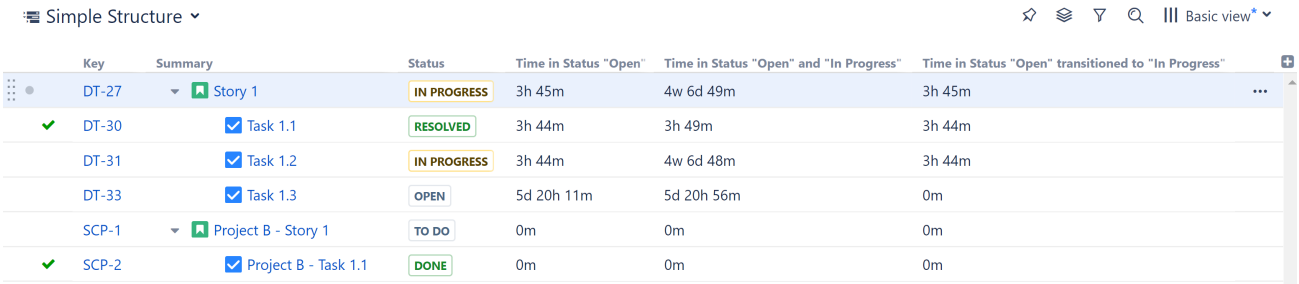
If the Sum over sub-issues option is unavailable for a given field, then the aggregate cannot be calculated.

Totals are Structure-Specific

The Total value for a specific issue may change depending on the selected structure, because its sub-issues can vary by structure.

3.7.1.13 Time in Status Column

The Time in Status column allows you to calculate how much time issues spend in a particular status. It can also track multiple statuses, so you can see how much time issues spend in a particular part of the overall workflow.



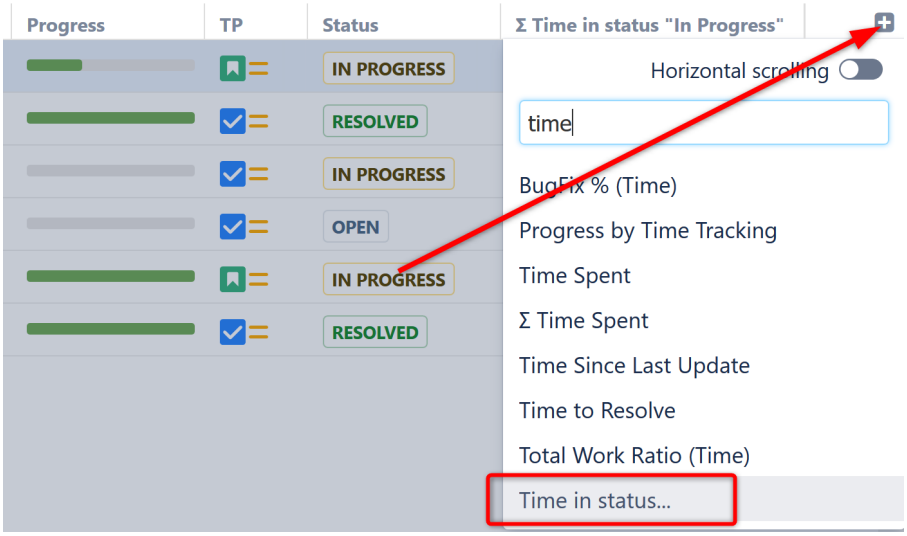
Each Time in Status column (you can include as many as you need in a structure) can be customized to display precisely the data you need to see.

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

https://www.youtube.com/watch?v=nODB0Dpoqks

Adding a Time in Status Column

To add a new Time in Status column, open the Add Column menu and search for **Time in Status...**

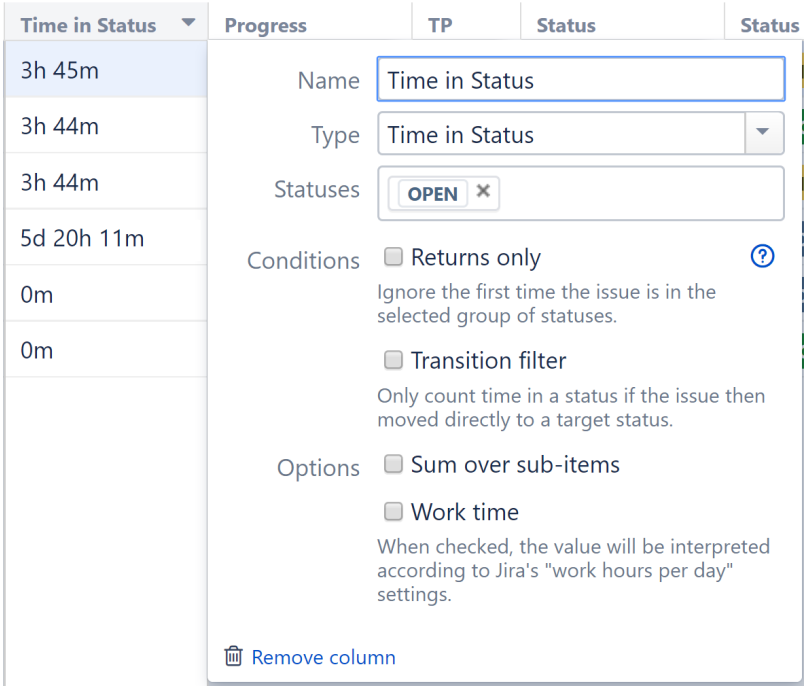


Once you've added the column to your structure, you can customize it to fit your specific needs.

Customizing a Time in Status Column

To customize a Time in Status column, open the column configuration panel (see page 332)and adjust any necessary settings:

- **Name** - we recommend giving the column a distinct name that alludes to the status(es) it's tracking, particularly if you have more than one Time in Status column in a structure.
- **Type** - this is the column type, and should remain unchanged, unless you wish to change an existing column to something else.
- **Statuses** - select which statuses and/or status categories to include when calculating time spent.
- **Returns only** - select this option to only track time when an issue returns to the selected statuses - see Returns only(see page 323) for full details.
- **Transition filter** - this option allows you to limit the results to only include time in a status when it directly preceded a move to a specific target status. For example, you could track just how long issues spend In Progress immediately before moving to Testing – if the issue moved from In Progress to any other status (back to Open, straight to Done, etc), that time would not be included.
- **Sum over sub-items** - select this option to get an aggregate total(see page 320) (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).

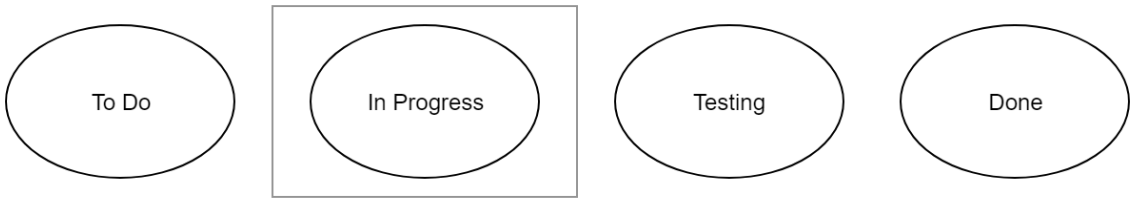


Returns only

When **Returns only** is selected, Structure will ignore the first time an issue is in the group of selected statuses/ status categories, and only track time if the issue returns to the selected statuses. This is a very useful way to track additional work.

Example 1 - A Single Selected Status

In the following workflow, you may want to see how much work has had to be re-done because issues are being passed along to testing too soon. To do so, under Statuses, you would select **In Progress**, and under Options check the **Returns only** box.

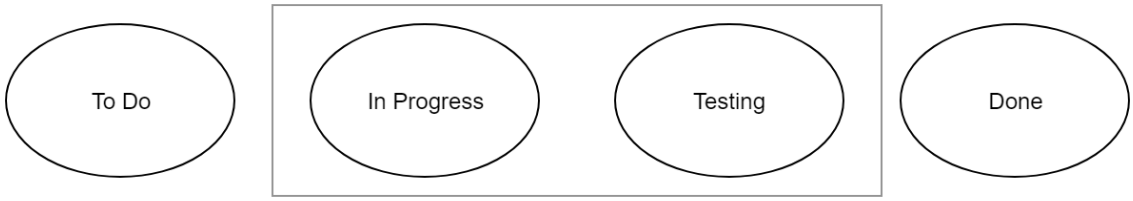


In this configuration, Structure would ignore the first time the issue was in the **In Progress** status, and only track time if it returned to the **In Progress** status after moving to Testing, Done, or back to To Do.

Example 2 - Multiple Selected Statuses

You may not have any concerns about items being returned to **In Progress** by your Testing teams, because that's just good practice - but you may be concerned with issues being returned to **In Progress** or **Testing** after they've been marked **Done**.

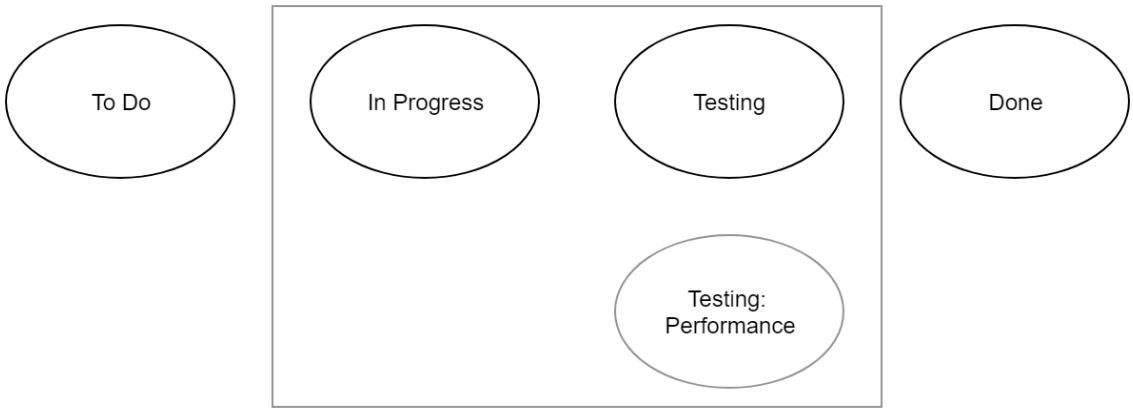
To do this, you could add both **In Progress** and **Testing** to your selected Statuses, and continue to select the **Returns only** box.



Now In Progress and Testing are considered a group to Structure - so moving from **In Progress** to **Testing** and then back to **In Progress** won't be considered a return (since the issue never left the group of selected statuses). But if the issue is moved to **Done** and then returns to either **In Progress** or **Testing**, that will be considered a return and Structure will track how much time is spent in those statuses after that point.

Example 3 - More Complex Setup

Many of us have multiple statuses for work in progress or testing. In this case, you can group all of those together (or just the relevant ones). In the example below, we've added **In Progress**, **Testing**, and **Testing: Performance** to our Statuses list.

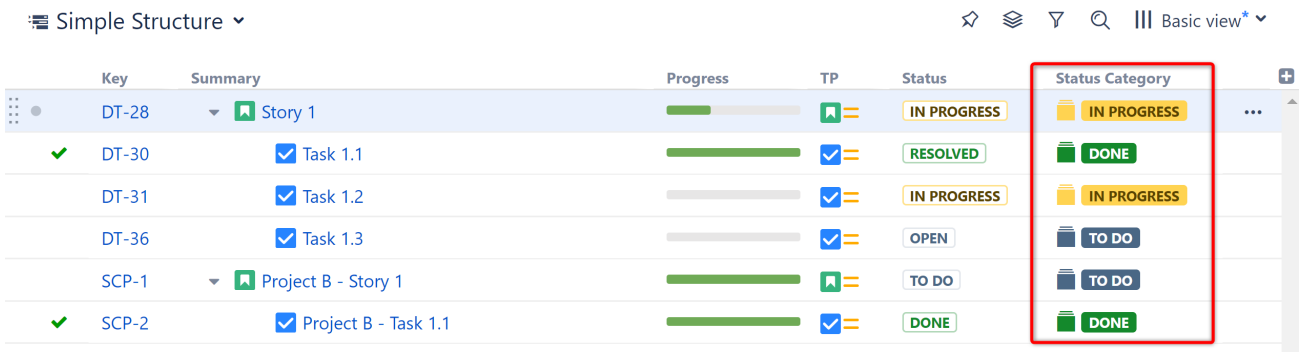


In this example, let's look at how Structure will calculate time if **Returns only** is checked:

- Issue moves from **To Do** → **In Progress** → **Testing** → **Testing: Performance** → **Done** - In this case, no time will be reported, because the issue never returned to a selected status.
- Issue moves from **To Do** → **In Progress** → **Testing** → **In Progress** → **Testing: Performance** → **Done** - Again, no time will be tracked. Even though the issue went back to **In Progress** a second time, it was continuously in one of the selected statuses, so this is not considered a return.
- Issue moves from **To Do** → **In Progress** → **Testing** → **Testing: Performance** → **Done** → **Testing: Performance** → **Done** - This time Structure will report the time spent the second time in **Testing: Performance**.
- Issue moves from **To Do** → **In Progress** → **Testing** → **Done** → **Testing: Performance** → **Done** - Structure will report the time spent in **Testing: Performance**. Even though the issue only reached this status once, it did so after leaving the selected statuses the first time; therefore, it is considered a return.

3.7.1.14 Status Category Column

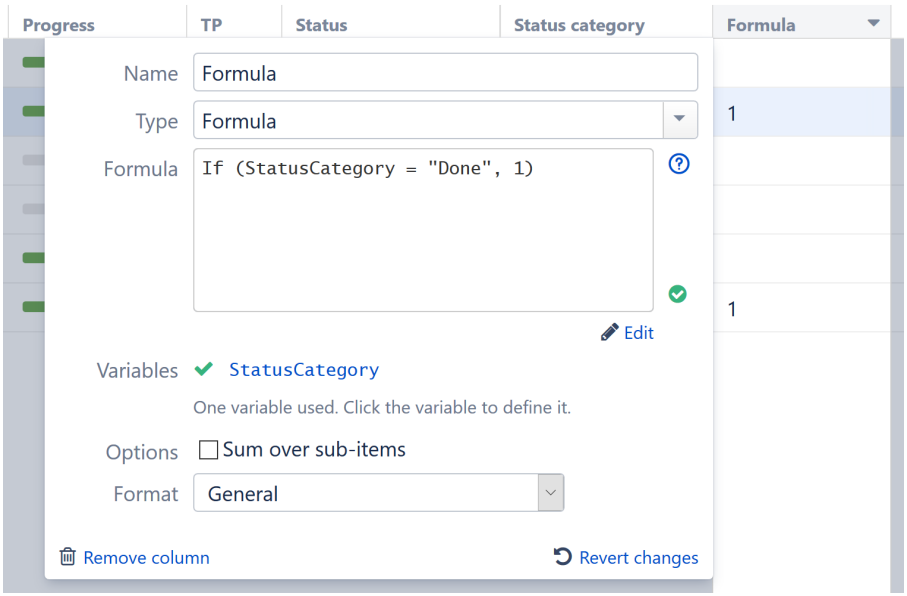
This column allows you to see at a glance which Status category each issue is in. This can be extremely useful if different teams/projects use different workflows or custom statuses.



Status category is a read-only column.

Status Category Values

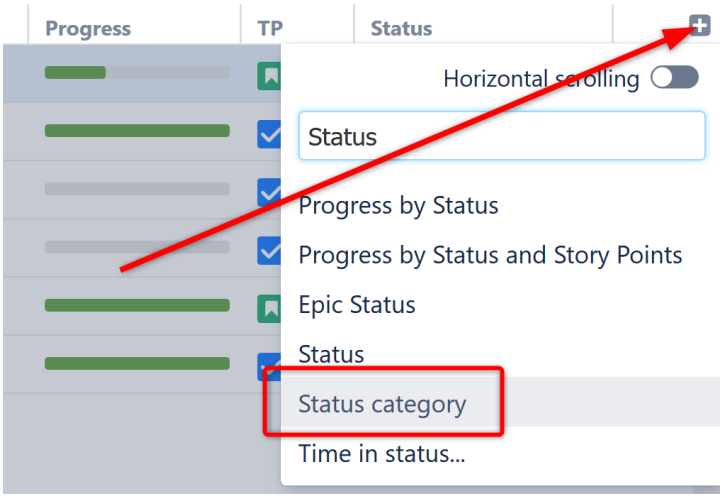
Status category values can be used to sort, filter or group issues. They can also be used in JQL queries and formulas(see page 197).



To group issues by Status category, use Group by Text Attribute.

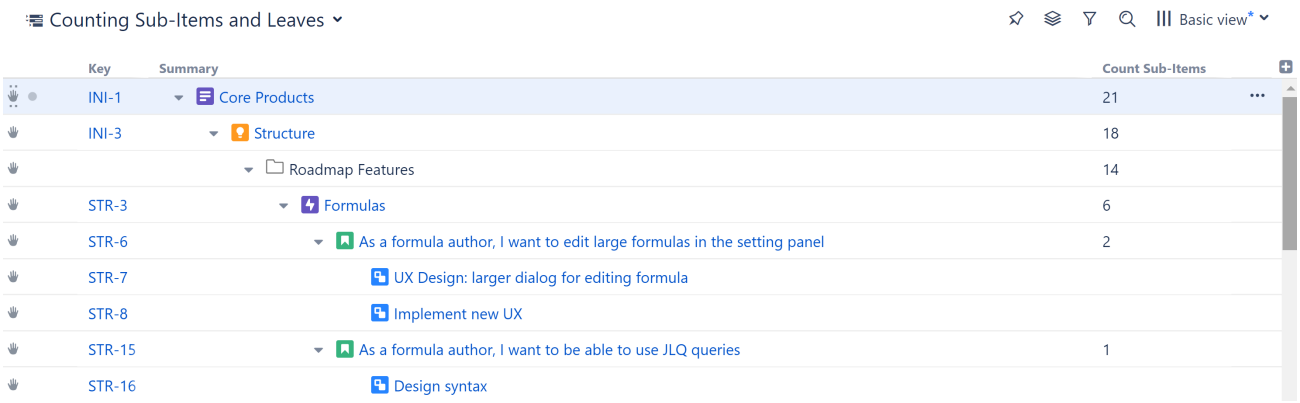
Adding the Status Category Column

To add the Status category column to a structure, open the Add Column menu and search for **Status category**.



3.7.1.15 Count Sub-Items Column

The Count Sub-Items column allows you to quickly see how many items are beneath an issue or folder within a structure.



In the example above, we can see that there are 2 sub-items beneath story STR-6 and a total of 14 sub-items contained in the Roadmap Features folder.

Use Cases

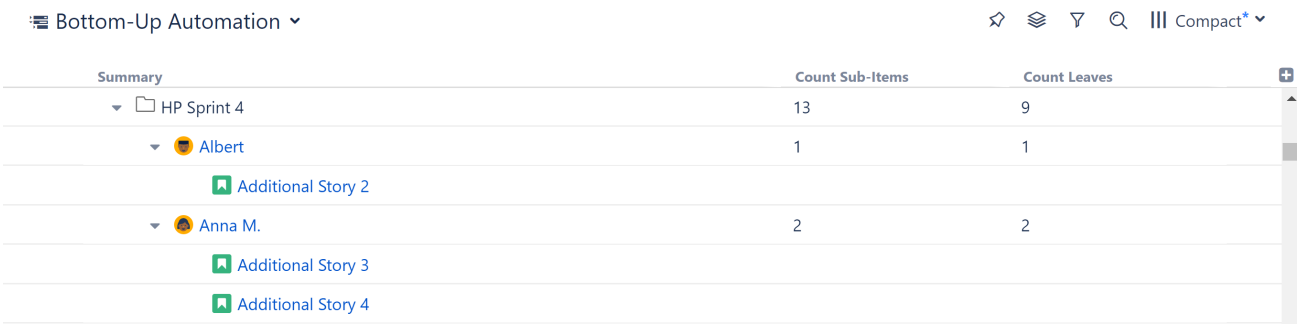
The Count Sub-Items column can be used to identify:

- How many items are contained within a folder or other grouping
- How many issues have been assigned to each resource (when your issues are grouped by Assignee)
- How many tasks are necessary to complete each story, epic, or initiative

Depending on how you've grouped your items, you may not want to count all sub-items. Instead, it may be more useful to just count the bottom level. In this case, you could use the Count Leaves column(see page 327).

3.7.1.16 Count Leaves Column

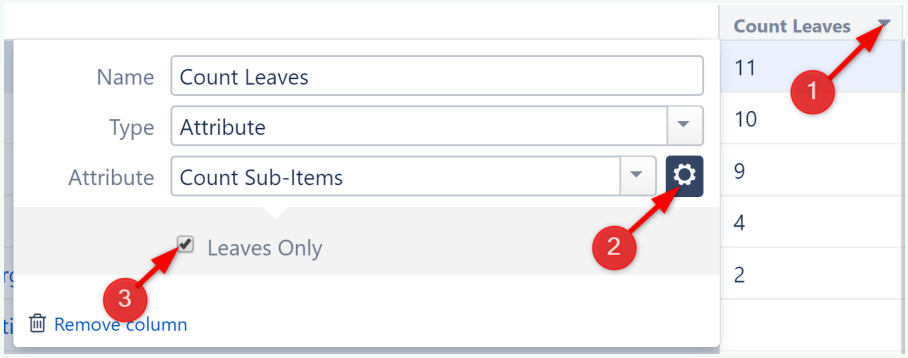
The Count Leaves column allows you to quickly see how many sub-items of an issue are at the last level of the hierarchy (they have no sub-items of their own). This is particularly useful when you have organized issues using Group generators(see page 141) or folders(see page 95) and you need a count of just the issues themselves.



Notice in the example above that HP Sprint 4 has 13 sub-items, but only 9 leaves. The Count Leaves column allows you to omit group items (in this case, the Assignee groups) and look just at the number of issues in the last level of the structure's hierarchy.

See also: Count Sub-Items Column(see page 326)

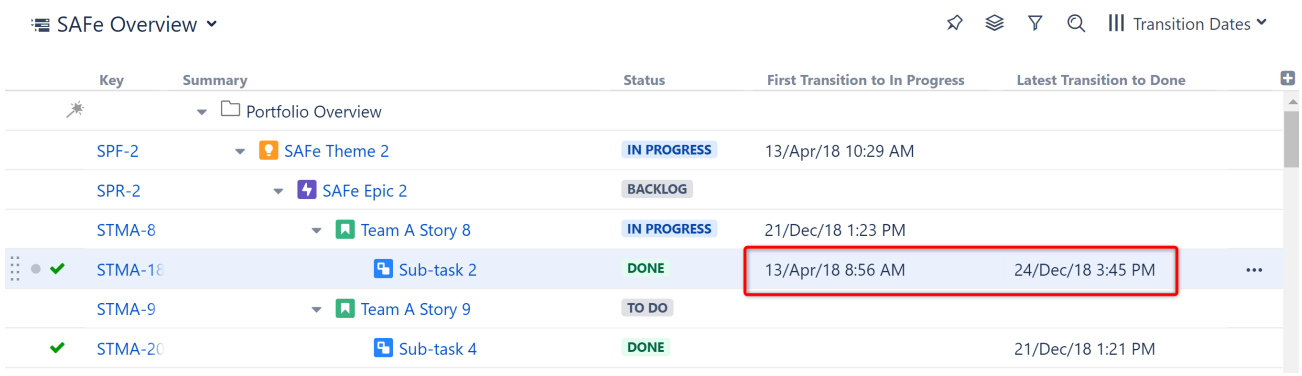
You can toggle between Count Sub-Items and Count Leaves. Open the column configuration, click the settings icon and check/uncheck **Leaves Only**.



3.7.1.17 Transition Dates

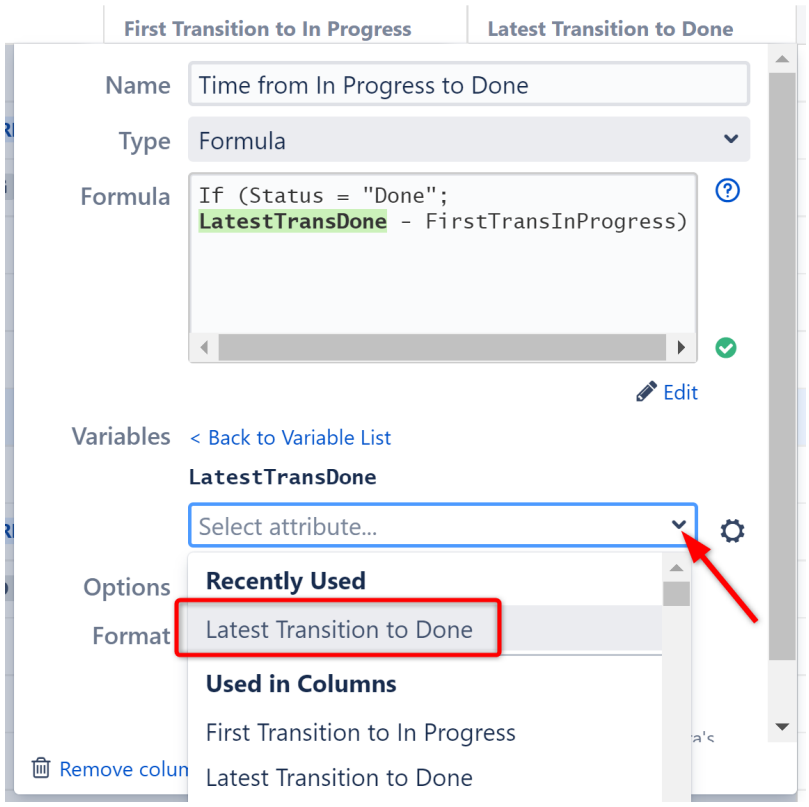
Transition Date Columns

The Transition Date (In Progress) and Transition Date (Done) columns allow you to visualize the dates issues were moved to In Progress or Done.



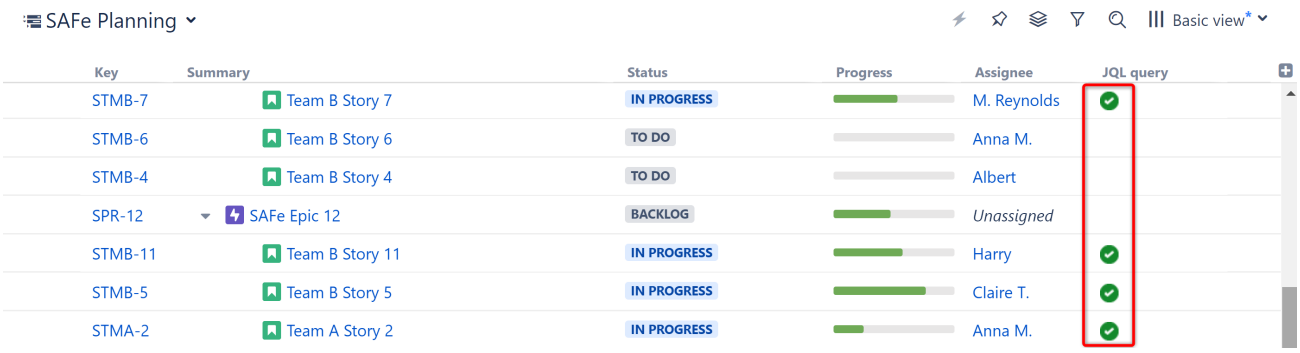
Transition Dates in Formulas

These Transition Date values can also be used in formulas by assigning a variable to the appropriate Transition Date attribute.



3.7.1.18 Query Match Column

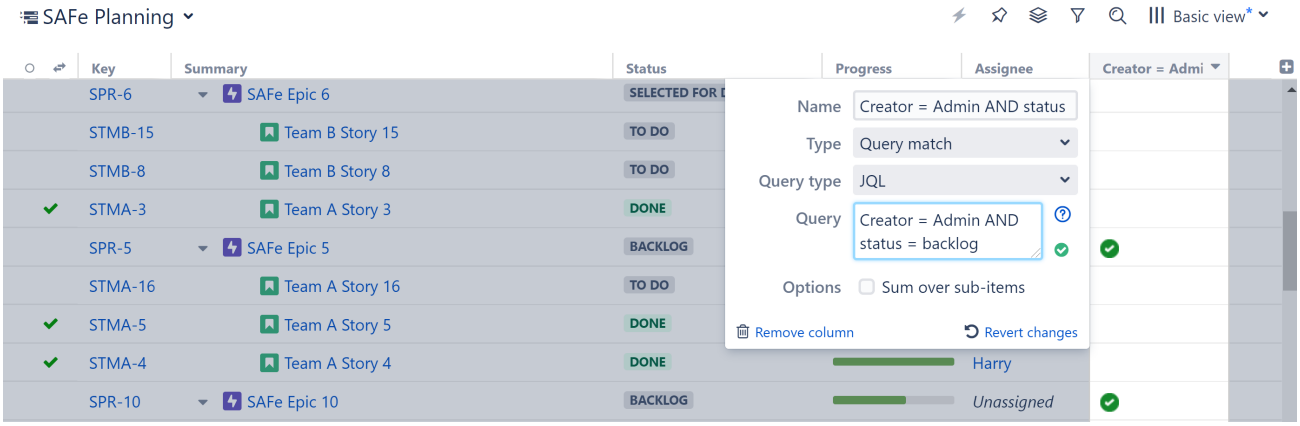
The Query Match column allows you to easily identify issues that match a specified text, JQL or S-JQL(see page 266) query. Items that match the query are noted by a check mark; those that do not are left blank.



Adding a Query Match Column

To add a Query Match column to a structure, click the **+** icon to the right of the column headers. In the Add Column dialogue, search for **Query match...**

In the Column Configuration panel, select the type of query (text, JQL or S-JQL(see page 266)) and enter your query. The name will update automatically based on your query, but you can also enter a custom name.

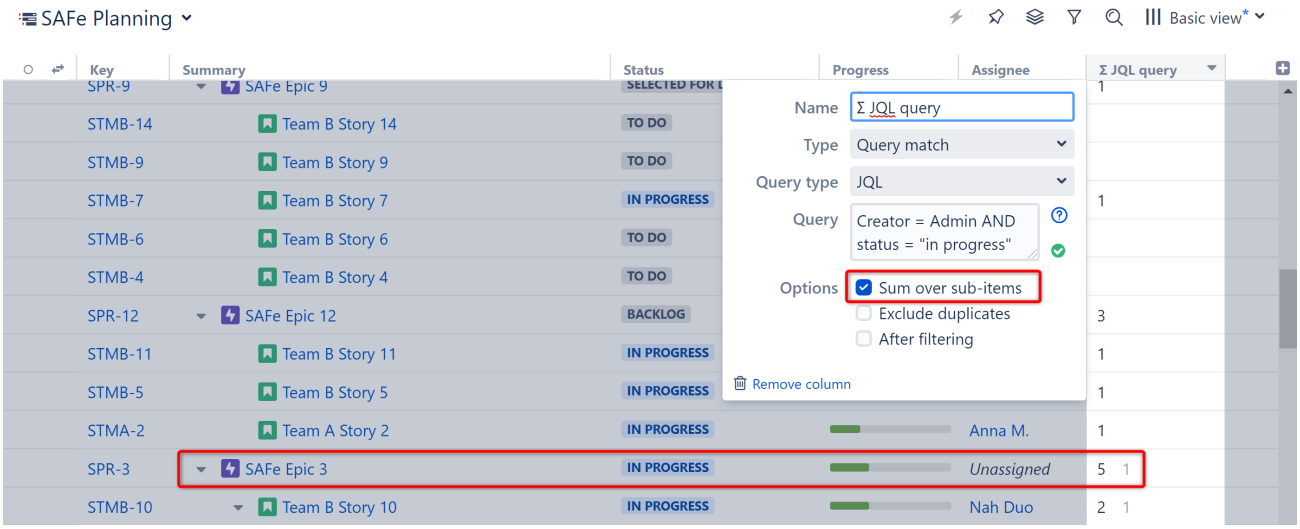


In our example, the Query match shows us any issues that were created by Admin and are still in the backlog.

Query match can also be used in formulas(see page 197). See Query Match as Variables(see page 206) 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.



In this example, Safe Epic 3 has 5 items (including itself) that were created by Admin and are in progress.

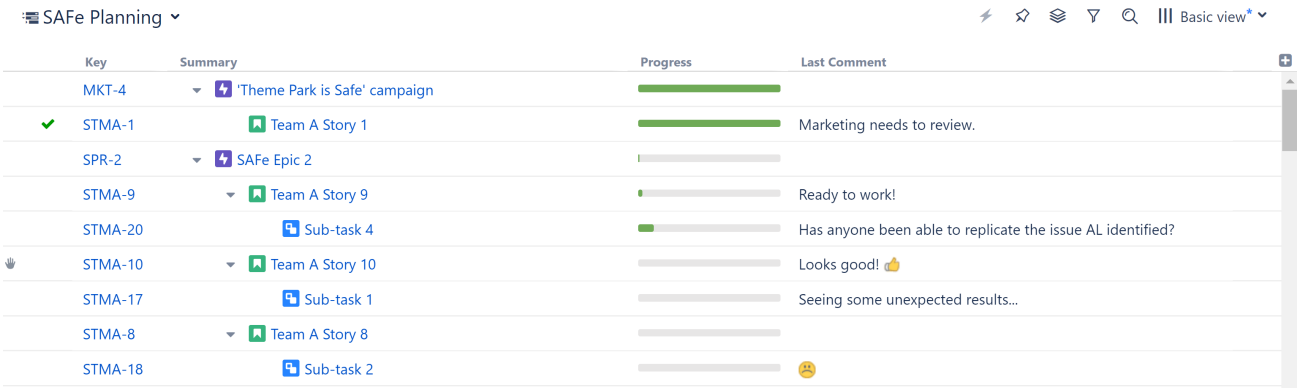
Aggregation Options

When **Sum over sub-items** is selected, you have a couple of options:

- **Exclude duplicates** - If an item appears more than once in the structure, it's value will only be included once within the aggregate total.
- **After filtering** - When checked, filtered items will not be included in the aggregate total. If this is left unchecked, the values of those items will be included in the calculation, even though they are not visible in the structure.

3.7.1.19 Last Comment Column

The Last Comment column displays the last comment left for each issue.



Adding this column provides a simple way to quickly review the most recent comments or questions left on every issue in the structure.

Permissions

Users will see the last comment they have permission to view. If a users does not have permission to see the latest comment, Structure will display the most recent they do have permission to view.

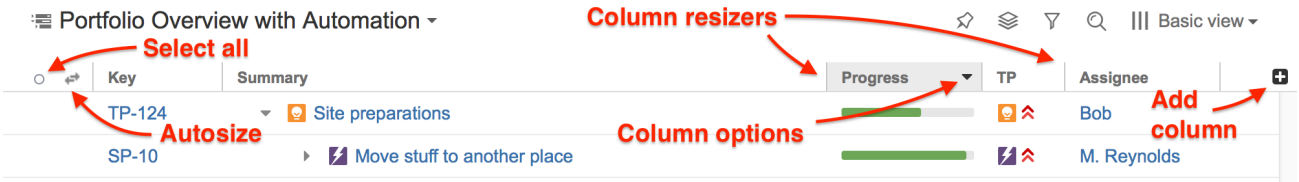
Last Comments in Formulas and Effectors

The Last Comments attribute can also be used in:

- Formula Columns(see page 197) - filter calculations based on latest comments, or include the latest comments as part of your results
- Effectors(see page 154) - write recent comments to a custom field

3.7.2 Customizing Columns

To configure structure columns, position the mouse pointer over the structure header. The grid controls should become visible, allowing you to select and resize columns.

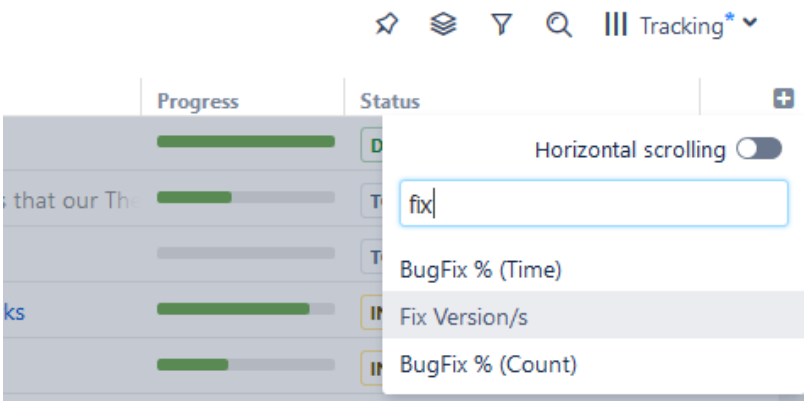


When you add, configure, remove or rearrange columns, you make adjustments to the *View* that's being used to display the structure. An adjusted view is marked with a blue asterisk (*). The adjustments are stored in your browser and affect only yourself – to make the changes permanent and available to others, you need to save the view or create a new view(see page 339).

3.7.2.1 Adding Columns

To add a column, click on the **+** button at the right corner of the table header. A drop-down with the available column presets appears. To select the desired column, you can:

- Use the mouse to find a specific column,
- Use the keyboard **arrow keys** to select the column and hit **Enter** when done, or
- Start typing the column name to filter the list, and then select the appropriate column.

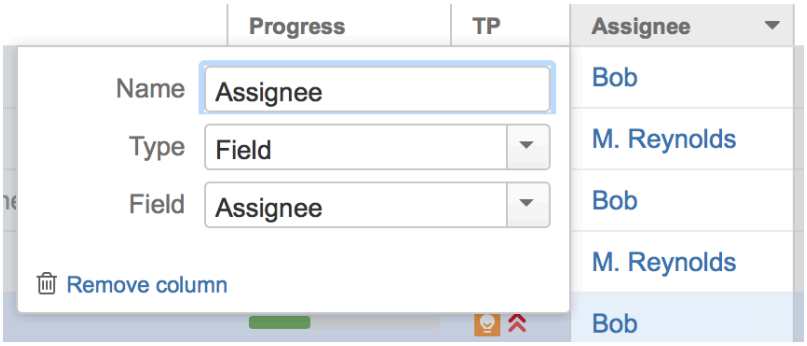


To abort adding a new column, hit **Escape**.

Use keyboard shortcut **TT** to quickly open the Add Column dialog (hit "t" twice).

3.7.2.2 Configuring Columns

To configure a column, click the arrow icon in the column header. The column will be highlighted, and its configuration drop-down will appear, allowing you to change the column name, type and other options.



The particular set of options available for the column is determined by its type(see page 295). For example, the Field(see page 297) column type lets you select the issue field to display and enable aggregation for numeric and time-tracking fields.

Any changes you make are applied immediately, so you can see the effect almost instantly. When you are happy with the column, simply close the configuration panel by clicking the arrow icon again or clicking anywhere outside the panel.

To cancel all of your changes use the **"Revert changes"** button at the bottom of the configuration panel. The column will be restored to its original state.

3.7.2.3 Removing Columns

To remove a column, click the arrow icon in the column header, then use the **"Remove"** button at the bottom of the column configuration panel.

You cannot remove the Summary Column(see page 296), Flags Column(see page 317) or Jira Actions Column(see page 317).

3.7.2.4 Rearranging Columns

You can change the position of a column by grabbing the column name with the mouse and dragging it to the left or right.

3.7.2.5 Resizing Columns and Autosize

Structure automatically tries to give all displayed columns enough space to display all information, but sometimes you might need to give more space to a column or two.

There are a number of ways to change column widths:

- **Grab the resizer and drag.** When you hover your mouse over a column, the resizer that is responsible for that column's width is highlighted. When the column size is close to what Structure considers ideal width (based on the displayed data), the resizer "snaps" to the perfect position.
- **Hold CTRL and drag resizer.** Works the same as above, but without snapping. You can use it to fine-tune column width.
- **Hold ALT (Option) and drag resizer.** In this mode, you will redistribute the space between two adjacent columns - increasing the width of one column and decreasing the width of another.
- **Double-click the resizer or the column header.** The column will automatically resize to the default size.
- **Click the "Autosize" icon (⇄) or double-click the Summary column.** All columns will be resized automatically, based on the displayed data.

If you are unable to comfortably view all of the columns or need to stretch columns beyond the visible panel, you may need to enable Horizontal Scrolling(see page 333).

Column widths are not part of the *View* and are not saved on the server or shared.

3.7.3 Horizontal Scrolling

Structure automatically adjusts column sizes to fit every selected column into the panel. While you can easily change the size of each column, by default you are still bound by the size of your Structure panel. This can make it difficult to view several columns at once or columns with a great deal of information (such as the Summary column or text columns), particularly when you're working with a Double Grid layout or viewing Structure on a small screen.

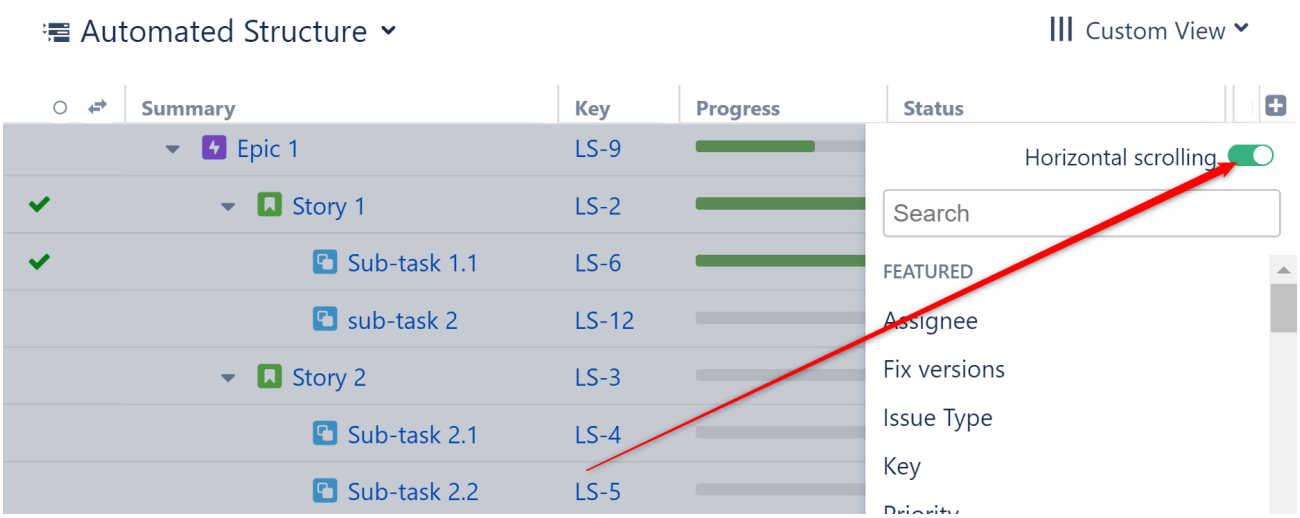
To make it easier to work with multiple columns and/or larger columns, you can turn on horizontal scrolling. This will allow you to work with as many columns as you need and set their sizes as large as necessary for easy viewing.

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

https://www.youtube.com/watch?v=nbhnE82UWsk

To enable horizontal scrolling:

- Click the Add Column icon "**+**"
- Click the **Horizontal scrolling** toggle.



Once you turn horizontal scrolling on, your view may no longer fit within the viewing panel. In this case, a horizontal scrollbar will appear at the bottom of the panel. Sliding this from left to right will allow you to bring different columns into view.

3.7.3.1 Pinned Columns

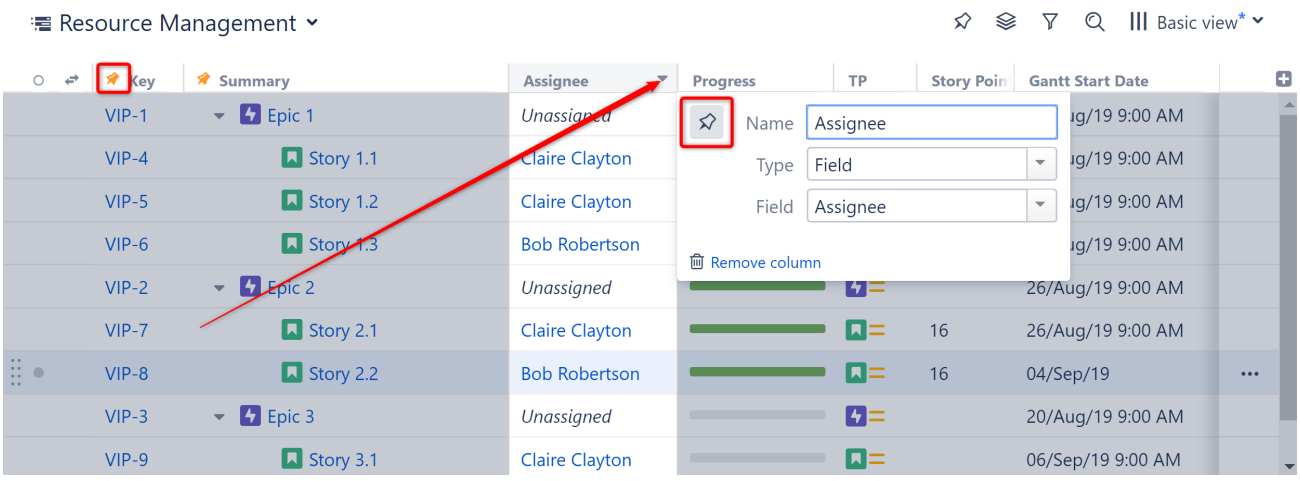
Not all columns move when you scroll. The Summary column(see page 296) remains visible at all times, so you can keep track of essential information as you focus in on other columns.

Additionally, the Jira Actions column(see page 317) is always available, though it may not be visible.

Pinning Additional Columns

To pin additional columns:

- Open the column's configuration menu and click the Pin icon. The column will be added to the pinned section in the farthest-right position.
- Drag the column to the left of any pinned column. The column will be added to the location you drop it.



Pinned columns will be moved to the left side of the screen and will remain visible as you scroll.

Unpinning Columns

To unpin a column:

- Open its configuration menu again and unclick the Pin icon. The column will be moved back to it's pre-pinned position.
- Drag it to the right of an unpinned column. The column will be placed wherever you drop it.

Additionally, when Horizontal scrolling mode is turned off, all pinned columns will be moved back to their original positions, when possible.

> The Summary column cannot be unpinned.

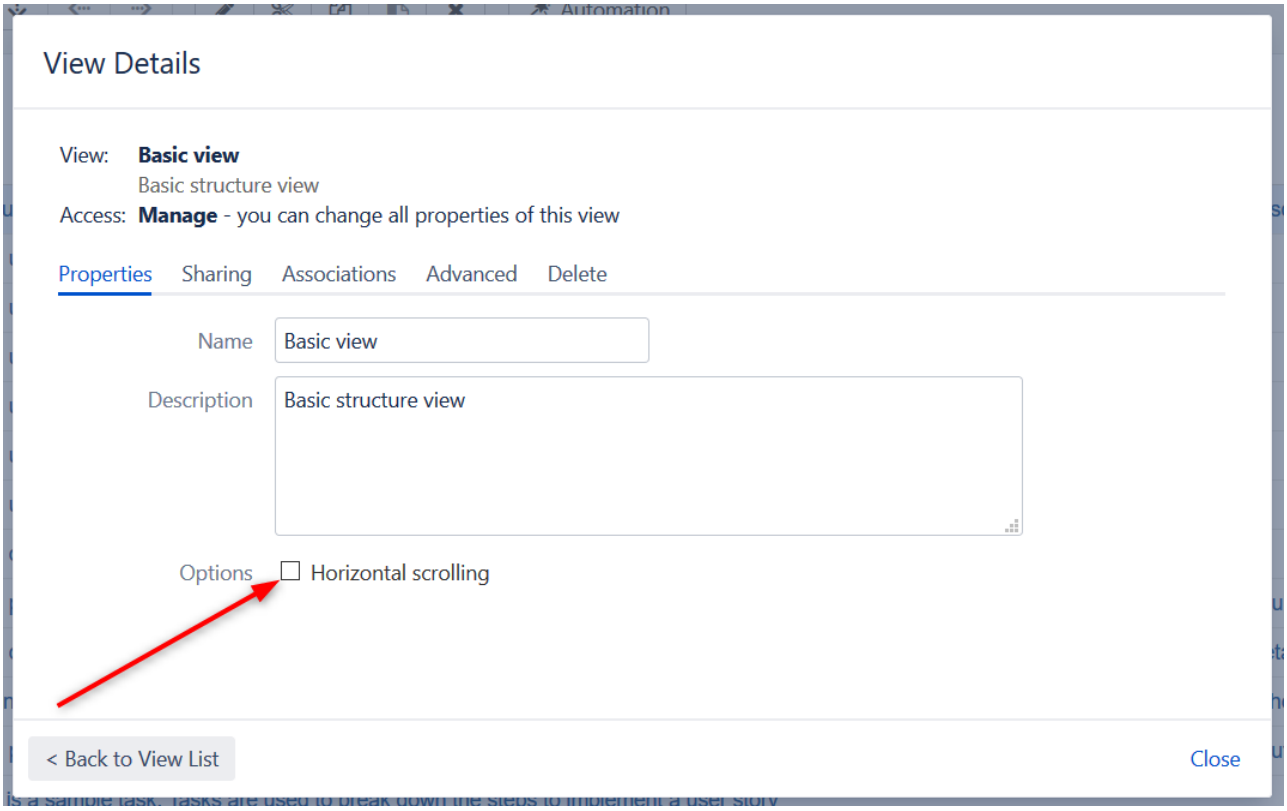
3.7.3.2 Saving Horizontal Scrolling Settings

Structure allows you to save your horizontal scrolling settings for a particular view.

For example, by default, the Basic View only shows a few essential columns. Keeping horizontal scrolling disabled for this view makes sure all of your essential information is visible at a glance. On the other hand, the Planning view often has several columns and might be easier to scroll through than trying to view all together.

To set your horizontal scrolling preference for a saved view:

- In the Views Menu, select **Manage Views**
- Locate the view you want to set and click **Details**
- Under the Properties tab, check or uncheck **Horizontal scrolling**



Horizontal Scrolling is enabled by default for the Planning and Tracking views. If you prefer not to use Horizontal Scrolling for these views, you can disable it using the methods above.

3.7.4 Configuring View

In Structure, a **view** defines which columns are displayed and in what configuration.

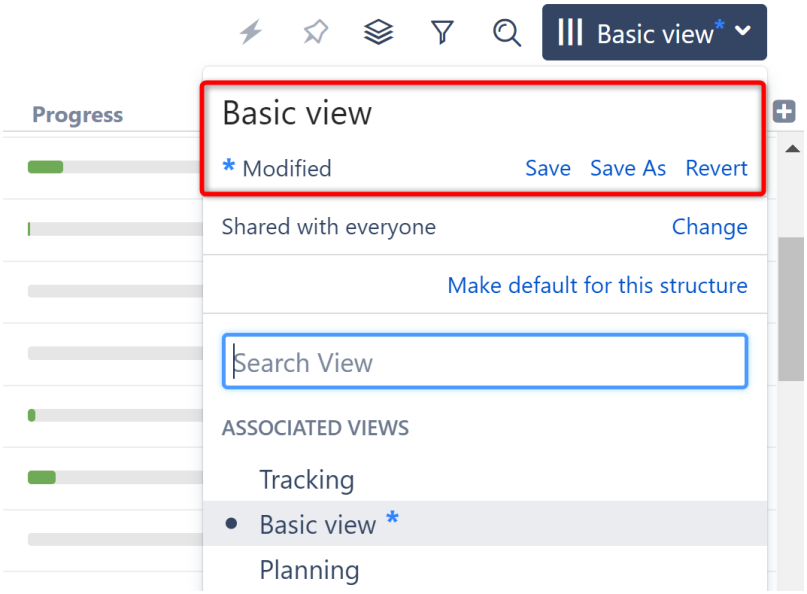
On the Structure Board, the current view is displayed in the top right corner. Click here to select a new view or save changes to the current view.

When viewing a structure from other locations in Jira, due to space limitations the view's name may not be displayed. To identify the current view, simply hover over the Views icon (the 3 vertical lines).

3.7.4.1 Changing the View

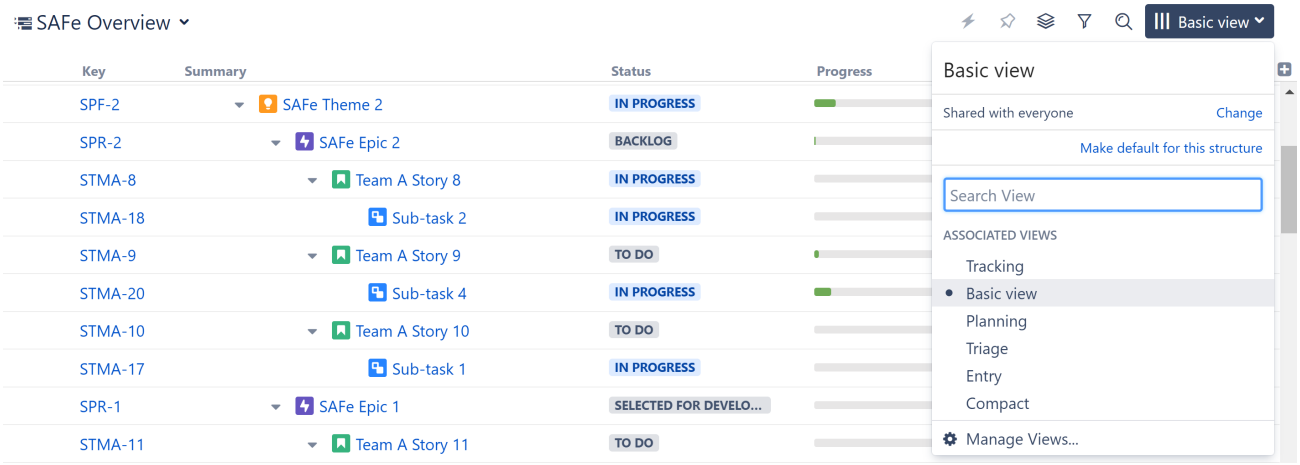
To change which columns are displayed or the order in which they appear, you can select a new view(see page 337) or manually add, remove or rearrange columns(see page 331).

When you manually change the column configuration, you create local adjustments to the currently-selected view. You can then save the changes (if you have permissions to change the view) or save and share your customization as a new view – see Saving and Sharing Views(see page 339).



3.7.4.2 Views Menu

The Views menu allows you to change the current view, save changes to a view, share a view, and manage existing views.



To open the Views menu, click the Views icon (the 3 vertical lines in the top-right corner of the Structure panel). Depending on where you're viewing the structure, the current view's name may be listed next to the Views icon.

In the Views menu, you will see the following:

1. The current view's name. Hover the mouse over the name to see the view's description.
2. If the view was modified, you'll see a corresponding message and links to save or revert these changes(see page 339). Additionally, in the Associated Views/Recent Views sections, a blue asterisk will appear next to the name.
3. Permissions settings and a link to change them.
4. The option to make the current view the default view (see page 339)for the structure, or an indicator that it is already the default view.
5. Search View. Search looks for any views that match the entered name, not only those in this list.
6. Associated Views list. This list can be customized for each structure by the structure owner or anyone who has Control access level (see page 362)for the structure – see Customizing View Settings(see page 360).
7. List of views you have recently used (excluding the views shown in the section above).
8. Manage Views link, which opens the View Management(see page 341) dialog.

Switching Views

To switch the current view, open the Views menu. From there, you have three ways to choose the new view:

1. Select it from the drop-down list. (This is the easiest method, but it may not include all available views.)
2. Search for it in the Search View panel.
3. Click **Manage Views...** to see all available views.

Switching View with Keyboard

You can also switch the current view using only the keyboard:

1. Use the **vv** shortcut to open the Views menu (hit "v" twice).
2. Use the arrow keys to select a view, or enter text to search for matching views.
3. Hit **Enter** to switch to a selected view or **Escape** to close the menu.

To go back to the previous view, use the **vvv** shortcut (hit "v" three times).

Opening a Structure

When a structure opens, it will select a view based on the history and settings for the location the view is being opened (Structure Board, Issue page, etc.). Structure will attempt to assign a view in the following order:

1. The last view you used for that structure in that location, if available.
2. The last view used for that structure in the opposite panel (primary panel vs. secondary panel(see page 351)) in that location, if available.
3. The default view (see page 339)for the structure, if available and viewing the structure on the Structure Board.
4. The global default view, as set by the Structure administrator.

The same structure opened on an Issue page will typically open with a different view then if it were opened on the Structure Board, because users need to see different information in different locations.

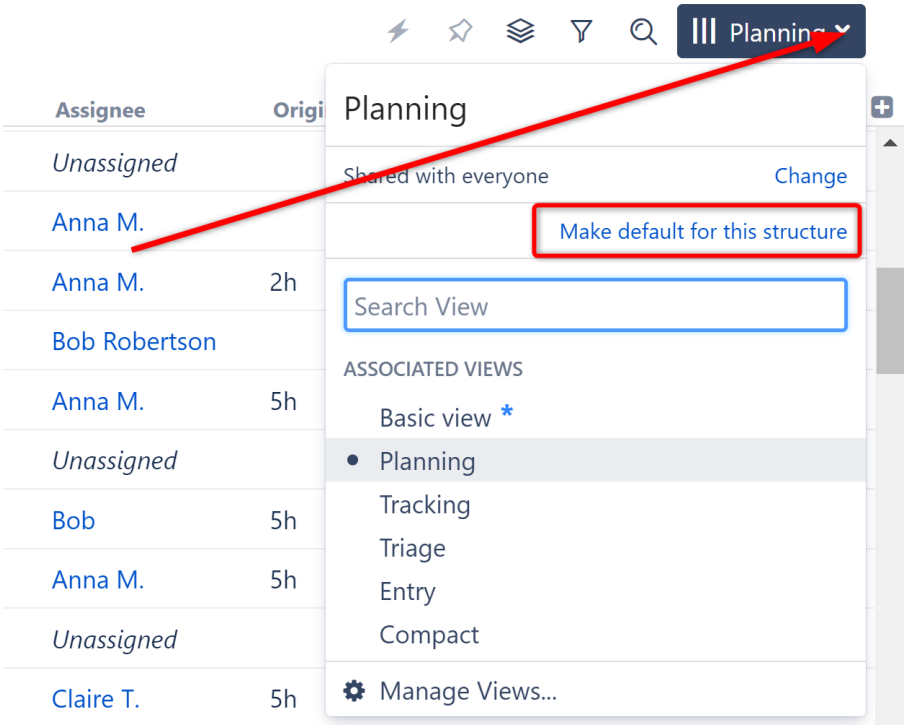
Default View

Users with Control permission(see page 362) can set a default view for the structure. When other users open a structure for the first time, they will see the default view. Additionally, if there is no recently-selected view available for the structure, it will open with the default view.

Setting the Default View

You can set the default view for a structure using Manage Structure (see page 360)or the Views menu. To set a default view using the Views menu:

1. Select the view you want to make the default. See Switching Views(see page 338) above.
2. Open the Views Menu again.
3. Click **Make default for this structure**.



If the selected view is already default for the structure, this section will let you know that you are either "Using the default view" or "Using the default view with modifications."

If you want to return to the previously selected view, use the keyboard shortcut: **vvv** (press "v" three times).

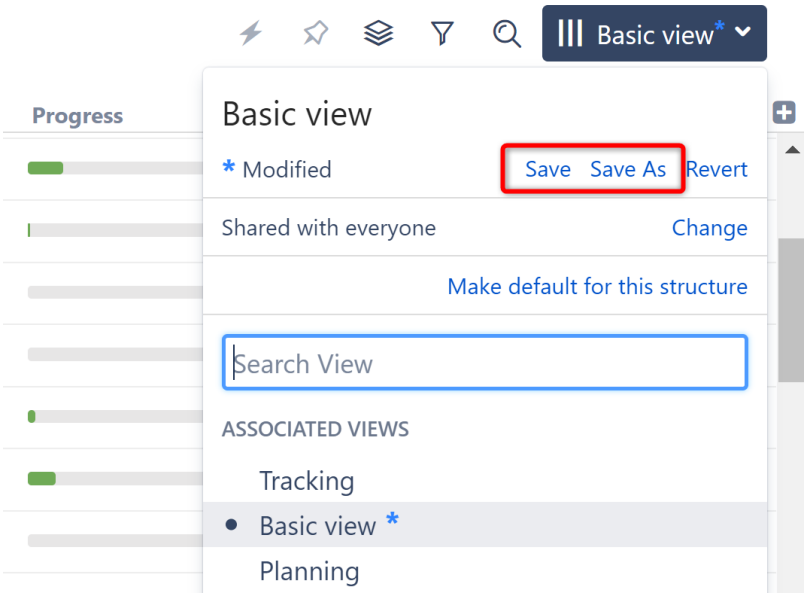
3.7.4.3 Saving and Sharing Views

When you add, remove or rearrange columns(see page 331), you are making changes to the currently-selected view. The view will now be marked with a blue asterisk, denoting that it has been modified from its original version.

These changes are stored locally and only visible to you. Anyone else using that same view will continue to see it in its original state, without your changes.

Saving View Adjustments

To make changes permanent and push them to other people using the same view, you need to save a new version of the view. To do so, open the Views menu and click the **Save** link. You can also select **Save As** to create a new view with the current changes, without affecting the original view.

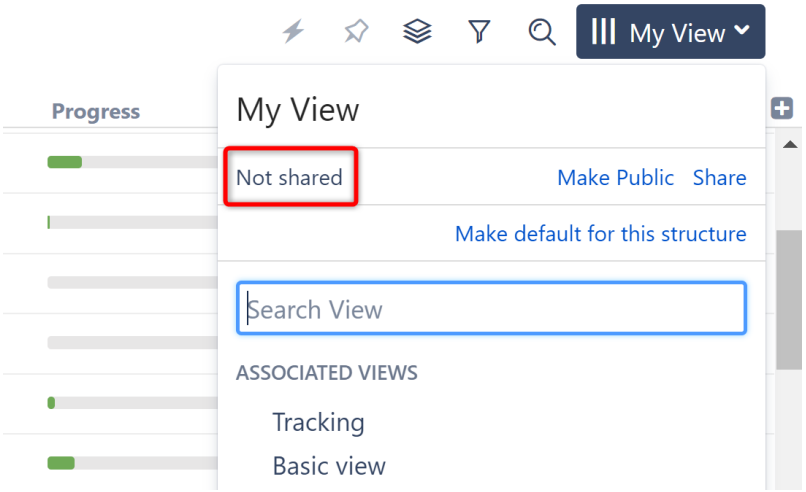


To save changes to an existing view, you need to have *Update* access level for that view (see View Sharing and Permissions(see page 346)). If you do not have permissions to change the view, you can still create a new view based on your modifications with the **Save As** link.

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.



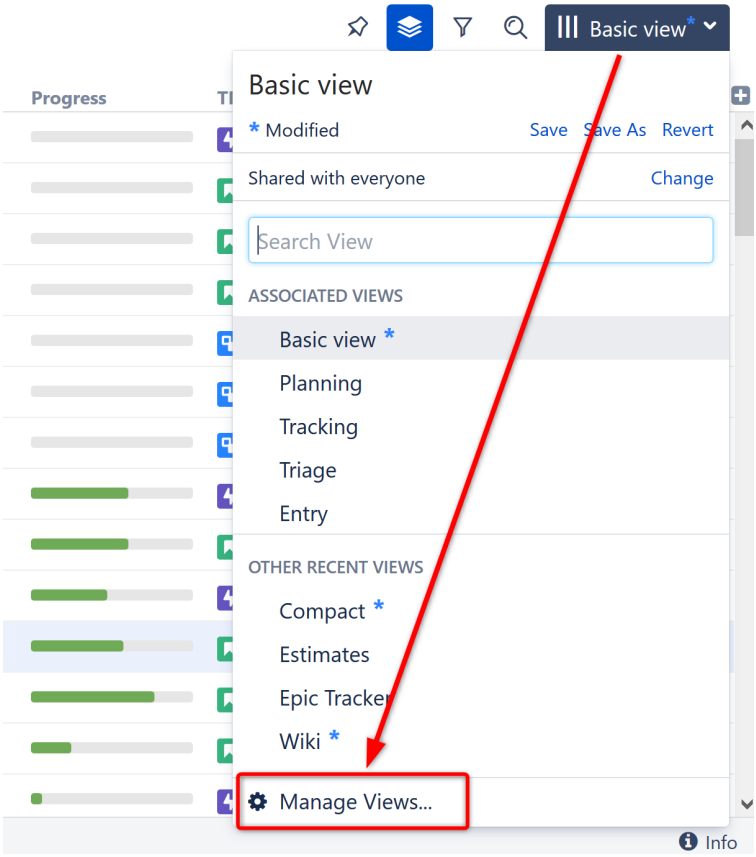
To share a view with other people, open the Views menu and:

- 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(see page 346) for details.

3.7.5 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(see page 337). For additional operations, or to browse all available views, open the Views menu and select **Manage Views**:

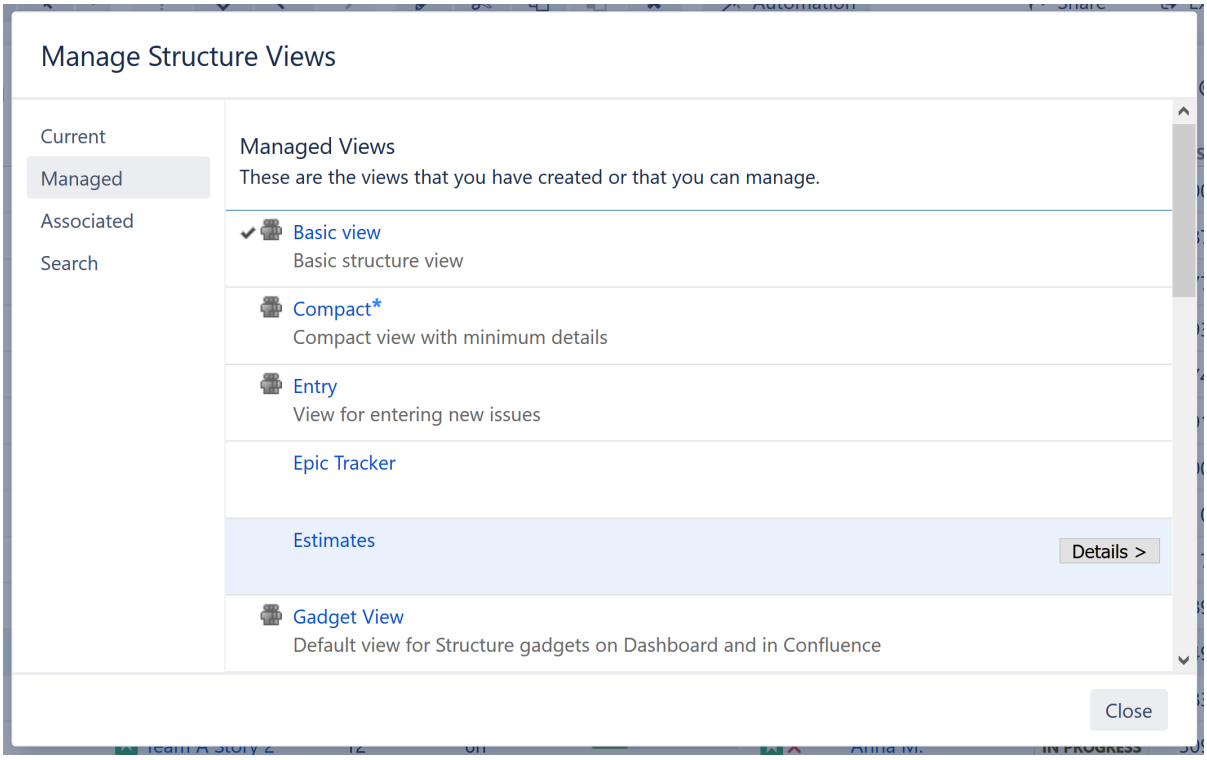


The following sections cover specific elements of view management:

- Locating a View(see page 342)
- Changing View Settings(see page 344)
- View Sharing and Permissions(see page 346)
- Associating Views with Structures(see page 348)
- Copying a View(see page 348)
- Deleting a View(see page 349)

3.7.5.1 Locating a View

The easiest way to locate a view is to find it in the drop-down list or search bar on the Views menu(see page 337); however, if you can't locate a particular view that way, or you need to make changes to a view, you can also locate views using the Manage Views dialog(see page 341).

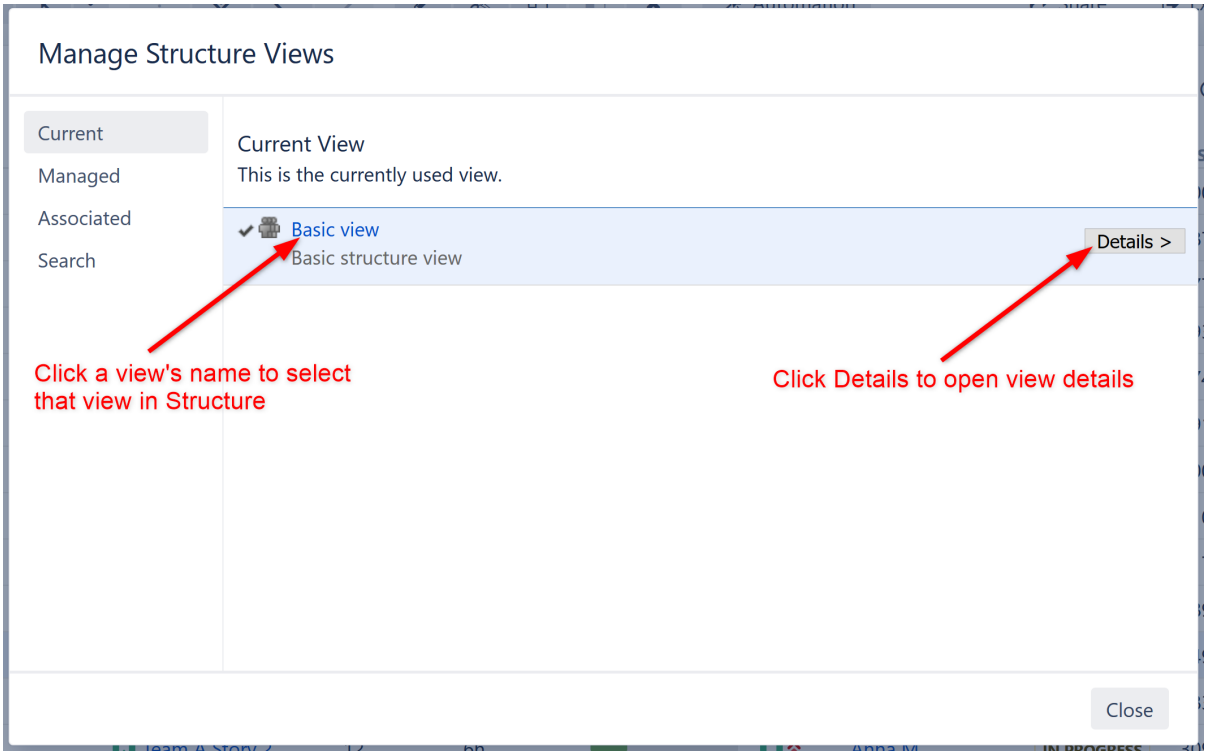


To find a particular view, select one of the following tabs:

- **Current** - displays only the view that is currently selected in the Views menu. You can quickly go to the current view's details from this tab.
- **Managed** - displays all views that you can **Manage** – that is, you have full administrative permissions(see page 346) for those views.
- **Associated** - displays all views that are associated with the currently-viewed structure (by the structure administrator).
- **Search** - allows you to search for views or display a list of all available views.

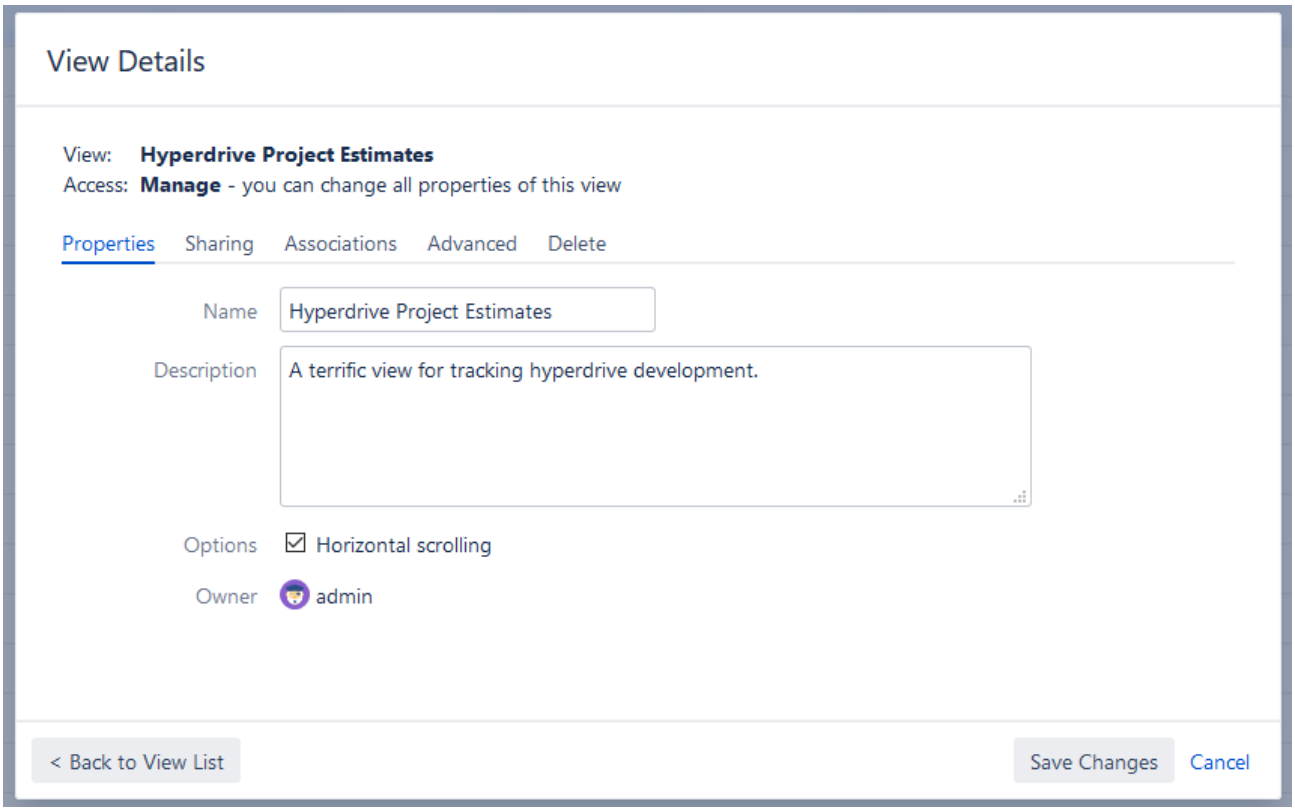
When you have located a specific view, you can click its name to switch to that view. The view will also appear in the **Other Recent Views** section of the Views menu(see page 337).

To see and edit View details, click the **Details** button that appears when you move the mouse pointer over the view record.



3.7.5.2 Changing View Settings

When you have located a view(see page 342) in the Manage Views dialog, click the **Details** button to open the View Details page in the same dialog:



The View Details page shows a number of tabs:

- **Properties** - lets you change the name and the description of the view, as well as select whether Horizontal Scrolling(see page 333) is enabled by default.
- **Sharing** - lets you view and modify sharing permissions for the view – see View Sharing and Permissions(see page 346).
- **Associations** - shows the structures which are associated with the view (have this view in their Views drop-down). See Associating Views with Structures(see page 348).
- **Advanced** - shows additional technical information about the view.
- **Delete** - lets you delete this view(see page 349).

⚠ The tabs and the scope of functionality available may be limited, depending on your access level to the view.

Renaming a View and Changing Other Properties

When you change a view's name, description, sharing permissions or anything on the Advanced tab, the changes are not saved until you click the **Save Changes** button. After you have saved the changes, they take effect for you and anyone else who has access to the view.

The Associations tab is different – it contains only links to structures. The associations between structures and views are managed by the structure administrator on the Manage Structure(see page 354) page.

3.7.5.3 View Sharing and Permissions

Like structures, views can be shared with different levels of access for each group of users.

There are four levels of access to a view:

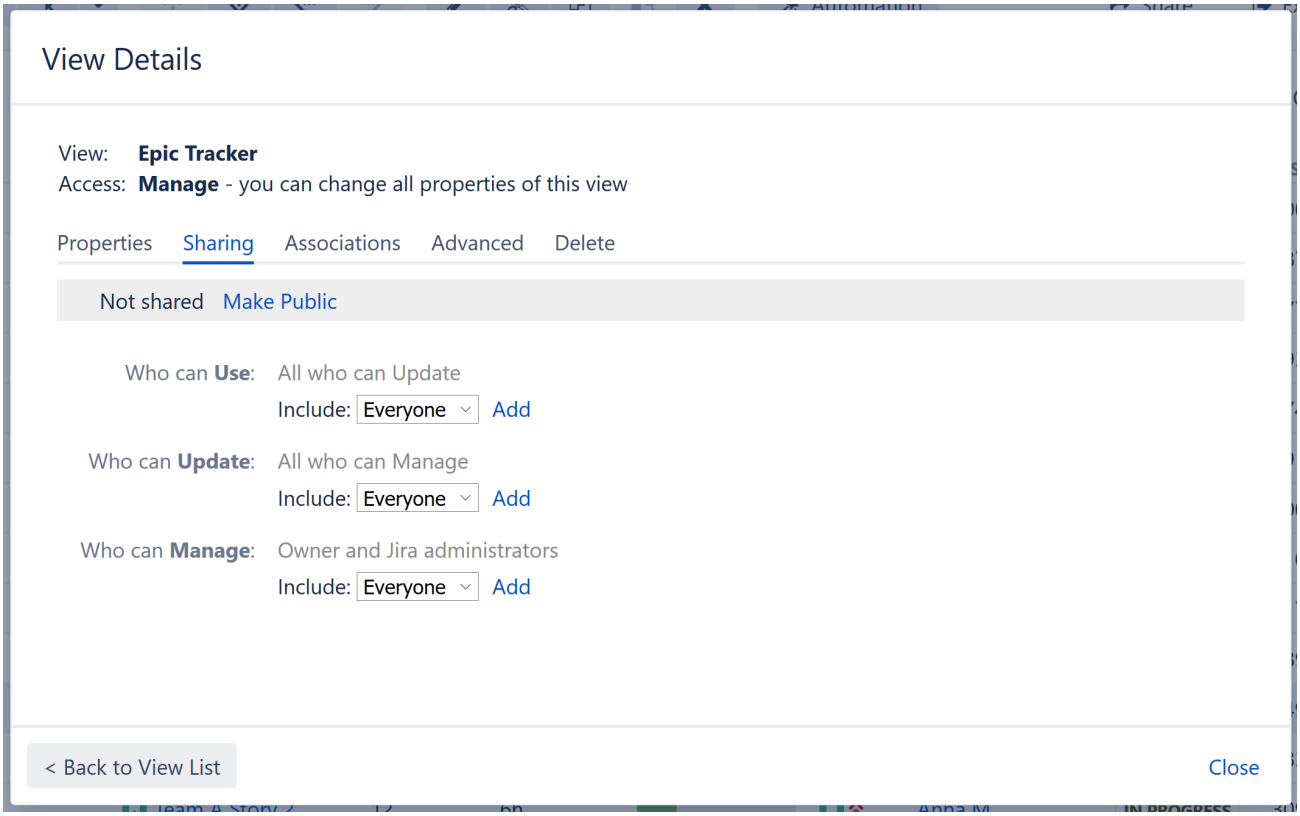
None	The view is not visible nor usable by the user.
Use	Read-only access: the user can use the view, but cannot modify it.
Update	The user can use the view, and also save view adjustments as the new version of the view. The user cannot modify view name or sharing permissions.
Manage	The user can change any of the view's properties and also can delete it.

View owner and Jira administrators always have **Manage** access to a view.

> People who have only **Use** permission for a view still **can** add, remove or rearrange columns(see page 331), but they can't save the modified configuration as a new version of the view. They can use the **Save As** link to create a new view with the modified configuration.

Changing permissions

If you have **Manage** access to a view, you can modify its permissions on the **Sharing** tab of the view details dialog.



For each level of access, you can define categories of users who have this type of access:

- Nobody
- Specific user groups
- Specific roles in specific projects
- Everyone (including anonymous users)

Higher-level access implies all lower-level access. So everyone who can **Manage** a view, can also **Update** and **Use** it - there is no need to add those users at all three levels!

Private and Public Views

When a view is not shared with anyone, it's called a **private view**. You can quickly make a view private by clicking the **Make Private** link – this will remove all permission assignments.

When **everyone** is given at least **Use** permission for a view, it is called a **public view**. You can quickly make a view public by clicking the **Make Public** link on the the **Sharing** tab and also in the Views menu(see page 337) – this will give **Use** permission for that view to everyone.

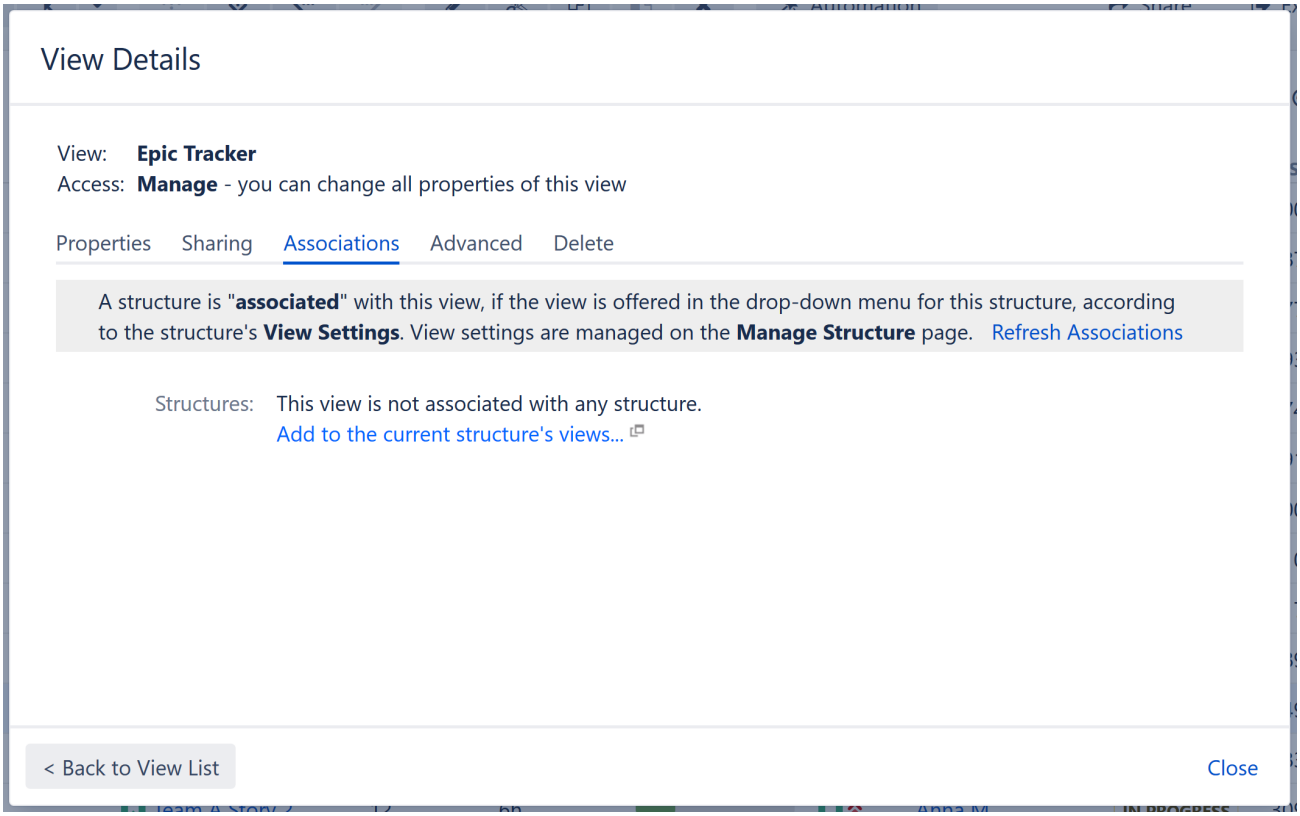
You need to have global **Create Shared Objects** permission to be able to share views.

3.7.5.4 Associating Views with Structures

Users with **Control** access to a structure can associate particular views with that structure. These views will appear in the **Associated Views** section of the Views menu(see page 337) when that structure is used.

> Jira administrator can also specify global default view settings(see page 482), which define associated views for structures that don't have customized view settings.

To view which structures a view is associated with, or to associate a view with the current structure, open the Manage Views dialog(see page 341) and locate the view to associate. Click **View Details** and select the **Associations** tab.



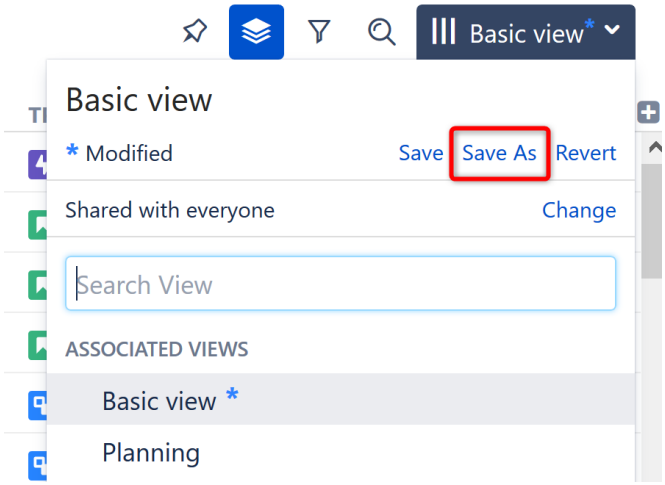
To associate the view with the current structure, click the **Add to the current structure's views...** link. If you have Manage access for the structure, this will open the View Settings(see page 360) for the structure and add the current view. Scroll to the bottom of the page and click **Apply**.

> View settings (associations between a view and a structure) are a property of the structure, not the view. The **Associations** tab on the View Details dialog is provided for convenience.

3.7.5.5 Copying a View

There's currently no way to directly copy a view, but you can use the **Save As** function to create a new view based on the existing view configuration:

1. On the Structure Board, select a view you'd like to copy, so it is your current view. You can use the Views menu(see page 337) or Manage Views dialog(see page 342) to find the view you need.
2. If you don't have local adjustments to the view, make some – for example, add a column, or change column order. (Note that just resizing a column does not change a view configuration.)
3. Open the Views menu and use the Save As(see page 339) link to create a new view.
4. Use the **Manage Views** dialog to review the new view's description and sharing permissions.



3.7.5.6 Deleting a View

To delete a view:

1. Open the Manage Views(see page 341) dialog.
2. Locate the view(see page 342) you'd like to delete and click the **Details** button.
3. Open **Delete** tab and click **Delete This View**.

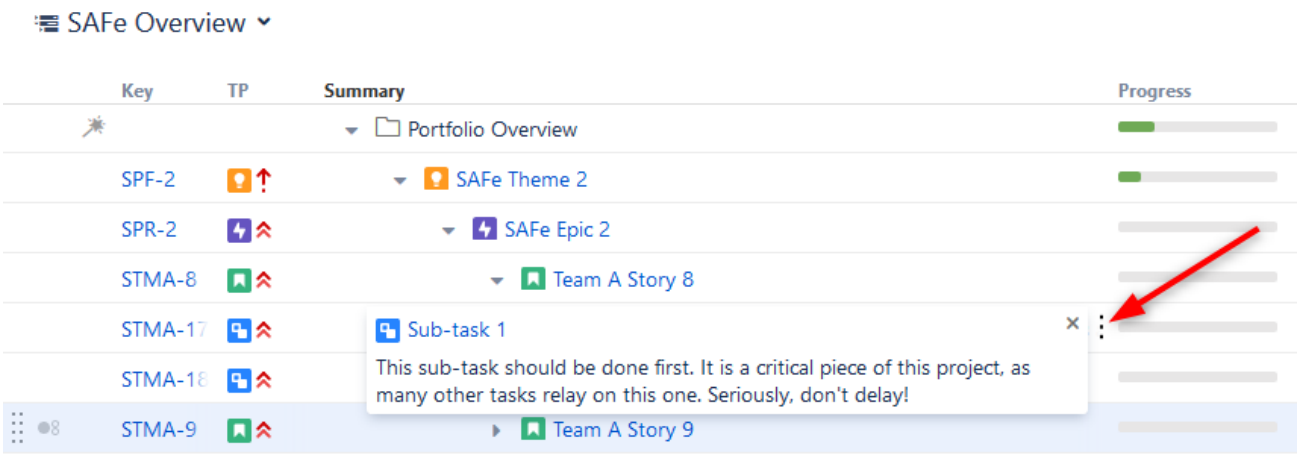
Deleting a view cannot be reverted.

Before deleting a view, open the Associations tab(see page 348) and review the list of structures that are associated with this view. The associations won't prevent you from deleting the view, but you may want to discuss the matter with the administrators of those structures.

3.7.6 Displaying Full Cell Content

In the Structure grid, if the content of a cell is larger than the cell's size, only a part of the content will be shown.

You can view the full content by clicking or hovering the mouse pointer over the "More" sign (three vertical dots) that appears at the right side of the cell.



To close the full-content panel, click the x, move the mouse away, press Esc or click anywhere outside the panel.

You can start editing the cell value even when the full-content panel is shown: double-click the panel or the Summary text in the panel.

3.7.7 Two-Panel Mode

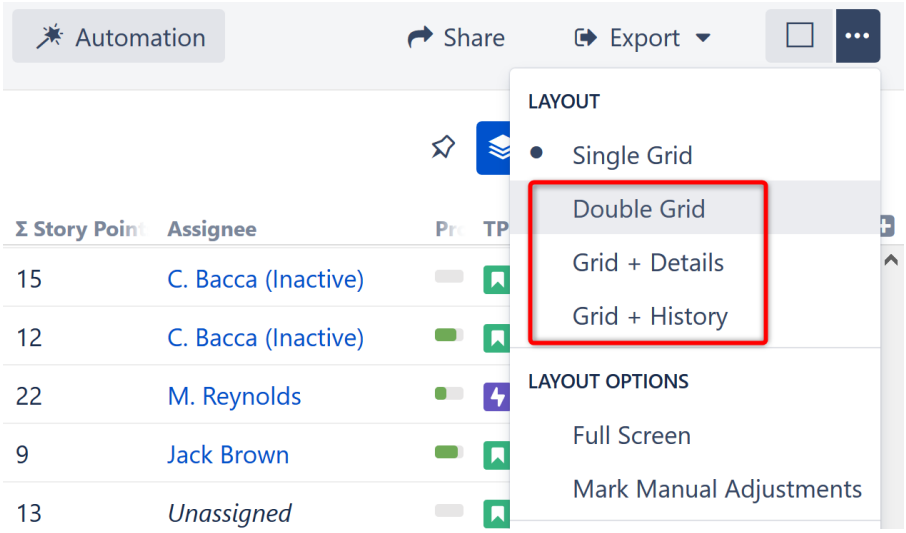
When working with a structure on the Structure Board, you can switch to the two-panel mode to take full advantage of the screen space.

While the left panel always displays the structure widget or search, the right panel can open any of the following:

- Another structure, so you can work with two structures side by side
- A text/JQL search
- Clipboard contents
- Issue details(see page 108) panel - as you click an issue in the structure, you can see/edit the issue details in the panel on the right
- History(see page 375) - see the list of changes done to the structure and navigate through them to see the previous versions of the structure
- Add-on Information:
  - With Structure.Gantt installed, you can view a Gantt chart in the secondary panel
  - With Structure.Pages installed, you can display Confluence page contents

3.7.7.1 Viewing the Secondary Panel

To open the Secondary panel, use the **Toggle Panels** menu in the Main Structure Toolbar(see page 76).



You can select from the following options:

- **Double Grid** opens the secondary panel with the structure widget. By default, the widget opens with the JQL search. You can switch to text search, clipboard or another structure by clicking the JQL label.
- **Grid + Details** opens the Issue Details Page(see page 108) for the currently-selected issue.
- **Grid + History** opens Structure History(see page 375).
- If you have additional add-ons that utilize the secondary panel, their options will be displayed below these.

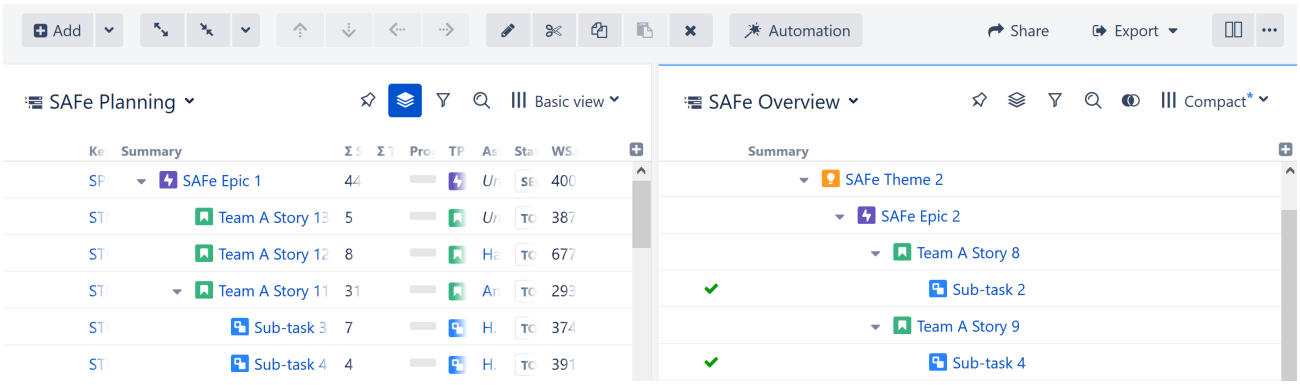
3.7.7.2 Resizing the Secondary Panel

You can divide the horizontal space between the secondary panel and the main panel by dragging the separating border.

3.7.7.3 Structure Widget on Secondary Panel

The structure widget that you open in the secondary panel is fully functional and differs very little from the widget in the main panel on the left. In both of them you can open structures, run JQL and Text search and open the clipboard.

Just like the main panel, it has its **panel toolbar** and the **Views menu**.



You can also use the Main Structure toolbar actions to work with the secondary panel widget. The toolbar actions will be applied to the panel that is in focus. The focused panel is highlighted with a thin blue line at the top.

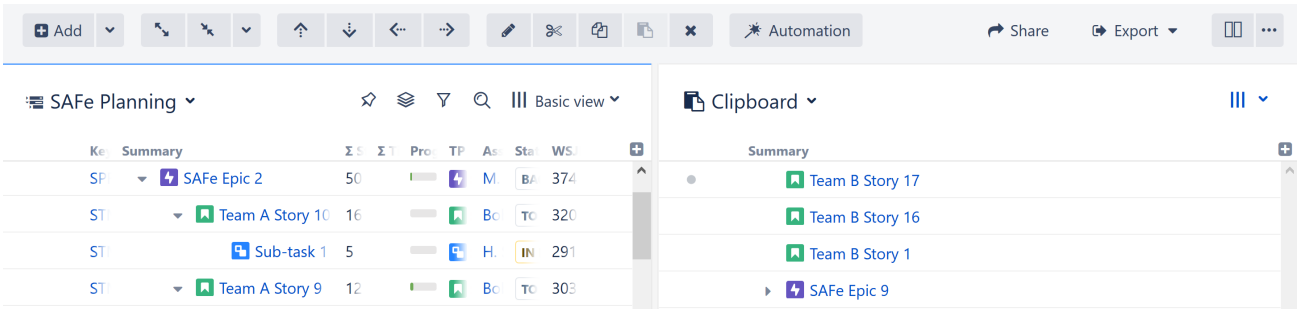
Hide/Show

The toolbar of the secondary panel has one extra function, Hide/Show. Clicking this button shows/hides items that are present in the main panel.

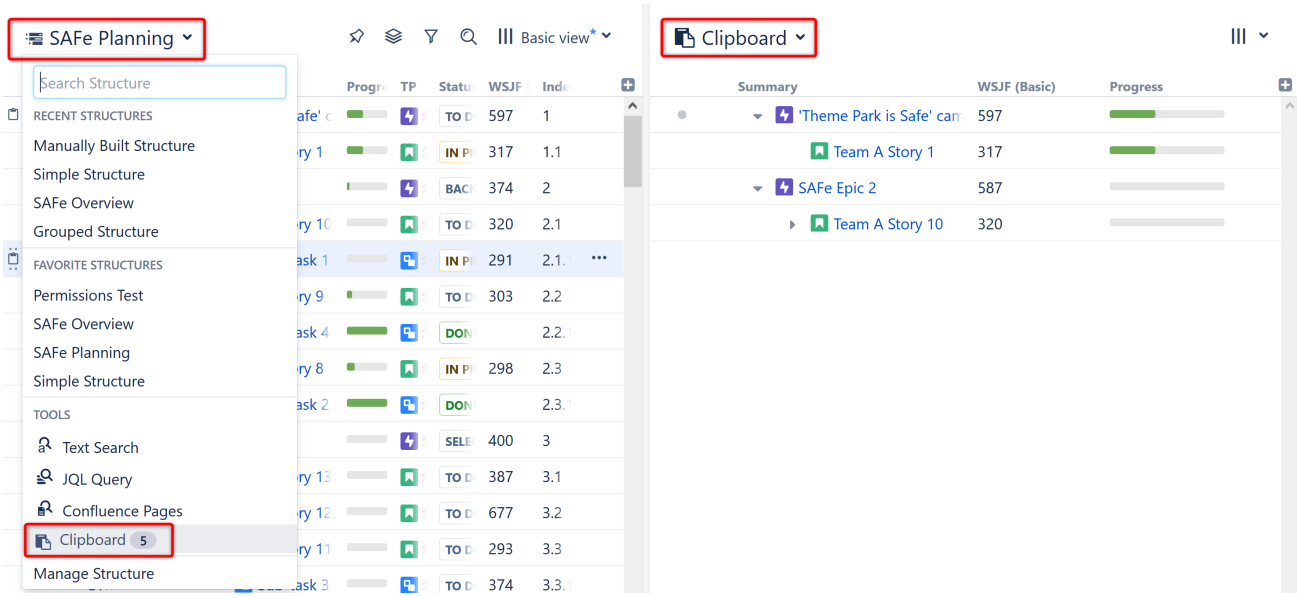
This is especially useful when you need to make sure the structure in the main panel has all the issues you've found using search in the secondary panel.

3.7.7.4 Issue Clipboard

The Issue Clipboard allows you to view issues that you have copied or cut. Clipboard contents can be viewed in the main Structure panel or the secondary panel.



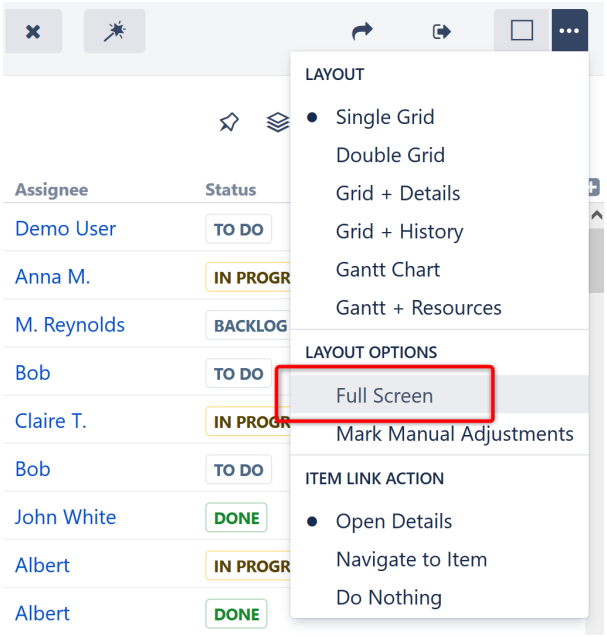
To view the Issue Clipboard, click the structure name or the search-type label at the top of the panel and select **Clipboard** from the menu.



To learn more, see Using Cut, Copy and Paste(see page 103).

3.7.8 Full Screen Mode

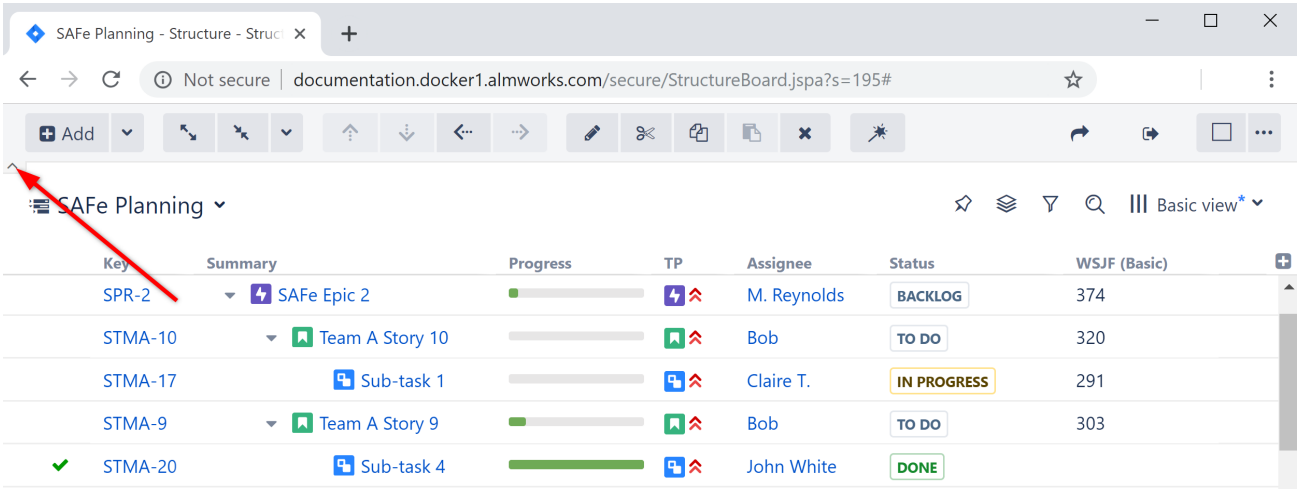
When working with the Structure Board you can turn on Full Screen mode to give more screen space to your data. Full Screen mode can be toggled using the **Toggle Panels** menu or by pressing Z on your keyboard.



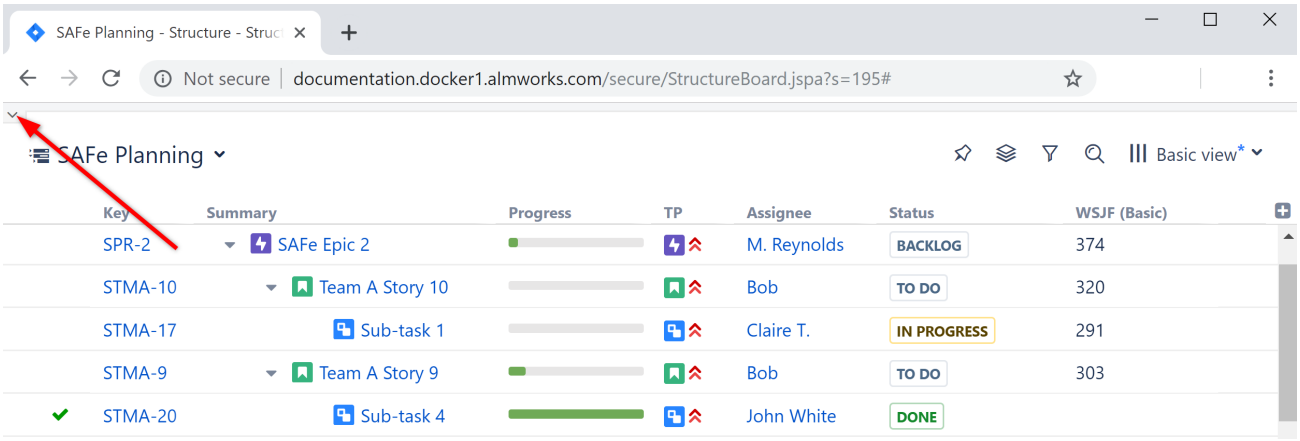
In Full Screen mode, the Jira application header is hidden and the Structure toolbar becomes more compact. To exit Full Screen mode, uncheck **Full Screen** in the Toggle Panels menu or press Z again.

3.7.8.1 Hide the Structure Toolbar

To save even more screen space, you can collapse the main toolbar by clicking the Collapse button.



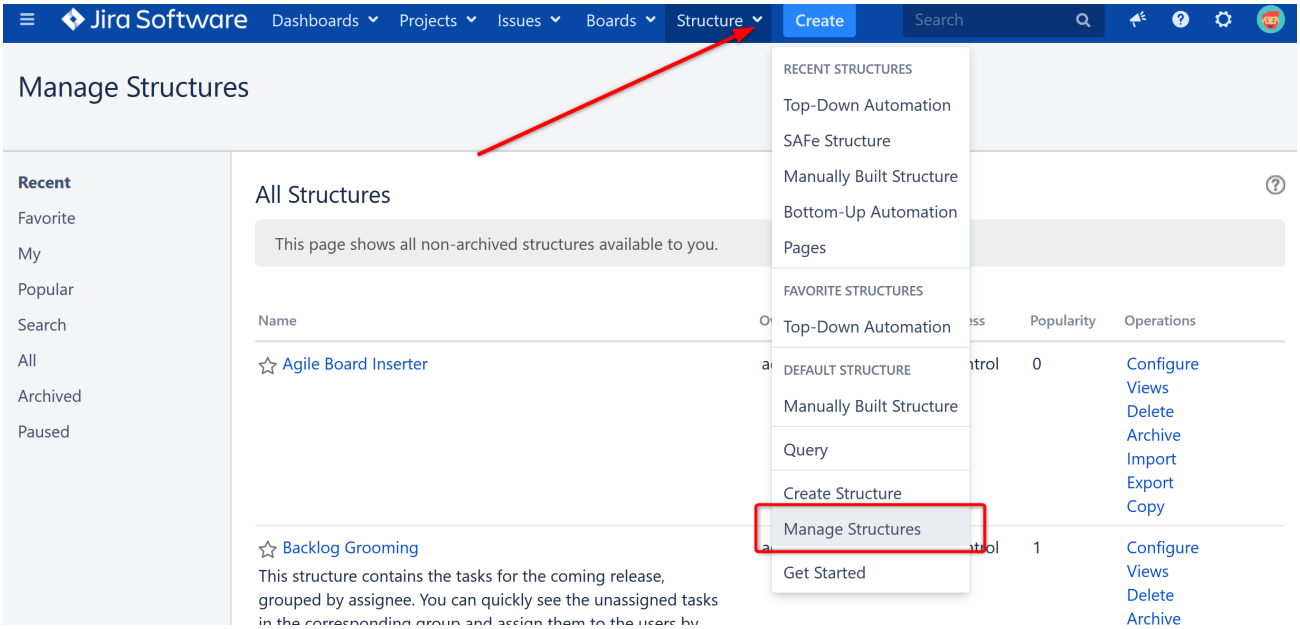
To use the toolbar in Collapsed mode, simply hover the mouse over the collapsed toolbar. It will reappear until you move the mouse away. To return the toolbar permanently, click the Expand button (where the Collapse button used to be).



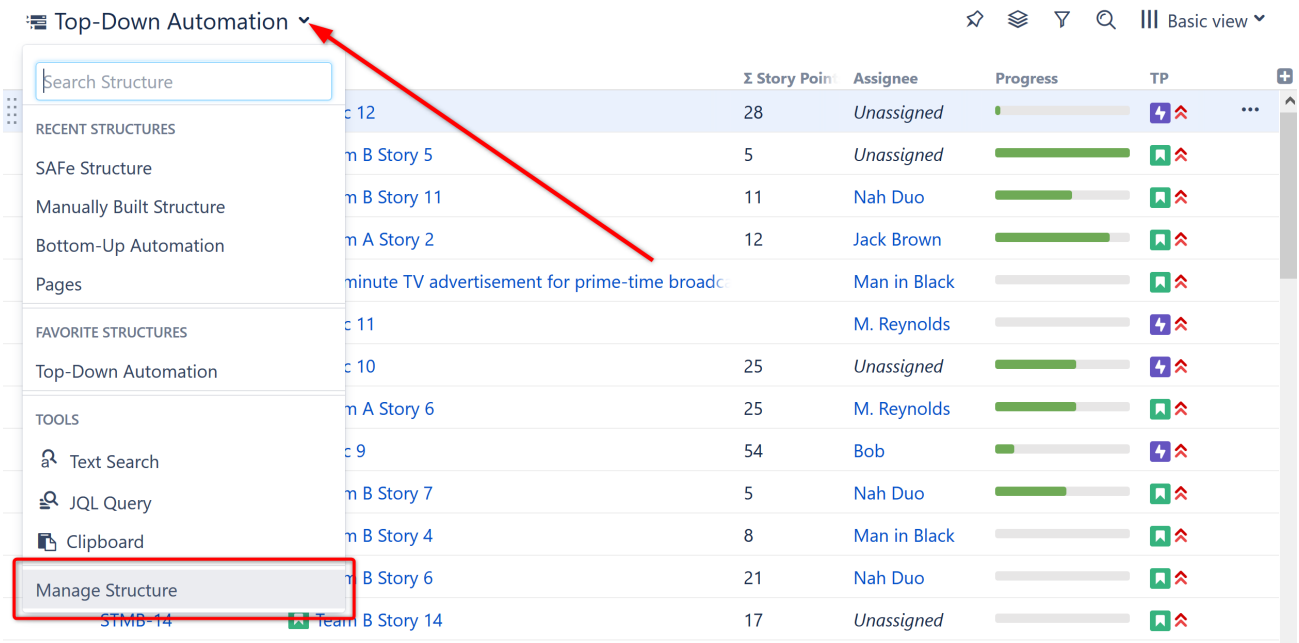
3.8 Managing Structures

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**.



You can also reach the Manage Structures page from the structure selector menu:



The **Manage Structures** page contains the following tabs:

- **Current** – shows the structure you are currently working with
- **Recent** – lists recently viewed structures, starting with the most recent/current structure
- **Favorite** – lists structures that you have marked as your favorite(see page 357)
- **My** – lists structures created by you
- **Popular** – lists structures that are marked as favorite by at least 2 users, ordered by their popularity(see page 357)
- **Search** – allows you to find a structure by name, owner or ID(see page 356)
- **All** – lists all structures visible to you
- **Paused** – only displayed when there are structures with Paused Automation(see page 354)

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.

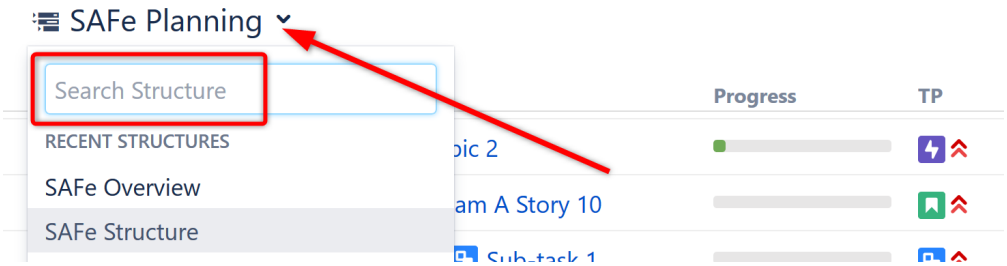
More about managing structures:

- Locating a Structure(see page 356)
- Default Structure(see page 357)
- Favorite Structures(see page 357)
- Structure Details(see page 359)
- Customizing View Settings(see page 360)
- Structure Permissions(see page 362)
- Creating New Structures(see page 365)
- Copying a Structure(see page 365)
- Archiving a Structure(see page 372)
- Deleting a Structure(see page 374)
- Template Structures and Projects(see page 374)
- Viewing History of a Structure(see page 375)
- Exporting Structures(see page 377)
- Real-Time Collaboration(see page 381)

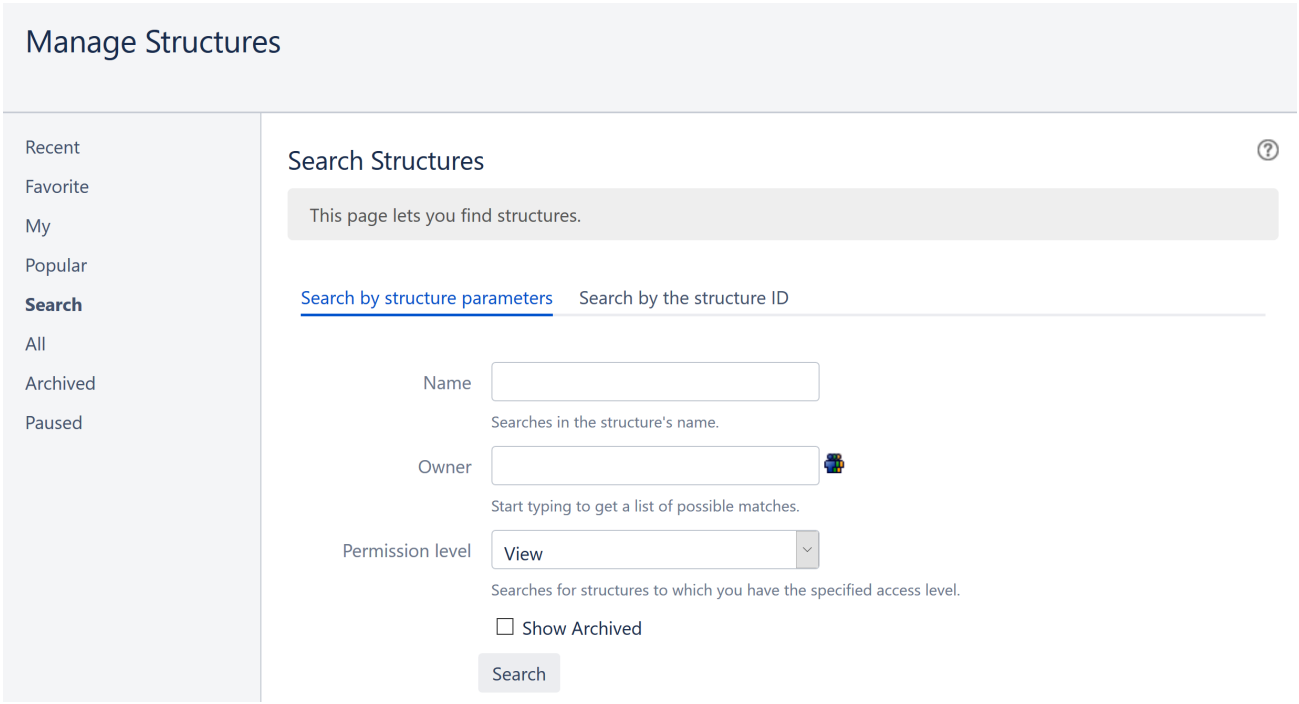
- Structure Activity Stream(see page 381)

3.8.1 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(see page 354). From the Jira menu, go to **Structure | Manage Structures** and select the **Search** tab.



3.8.1.1 Finding Structures by Name, Access Level or Owner

To search for structures by their properties:

1. Enter any of the search parameters. Parameters are:

Name	Only structures that contain the specified text in their name will be shown. If you don't know the exact name, you can use a part of a word that you know should be in the structure's name.

Owner	Only structures that are owned by the specified user will be shown. *Note: This only works if you have permission to browse users.*
Permission Level	Lets you select the structures that you can Edit or Control, according to the selected permission level. (For example, if you select **Edit** permission level, you will see all structures that you can edit and control, but you will not see structures that you can only view.)

2. Click **Search** without entering any parameters. All structures visible to you will be shown.

3.8.1.2 Finding a Structure by Its ID

To perform a search by a structure's numeric ID:

- Click the **Search by the structure ID** tab.
- Enter the structure ID. (It must be a number.)
- Click **Search**. If there's a structure with the specified ID and you have the permission to view it, it will be shown.

3.8.2 Default Structure

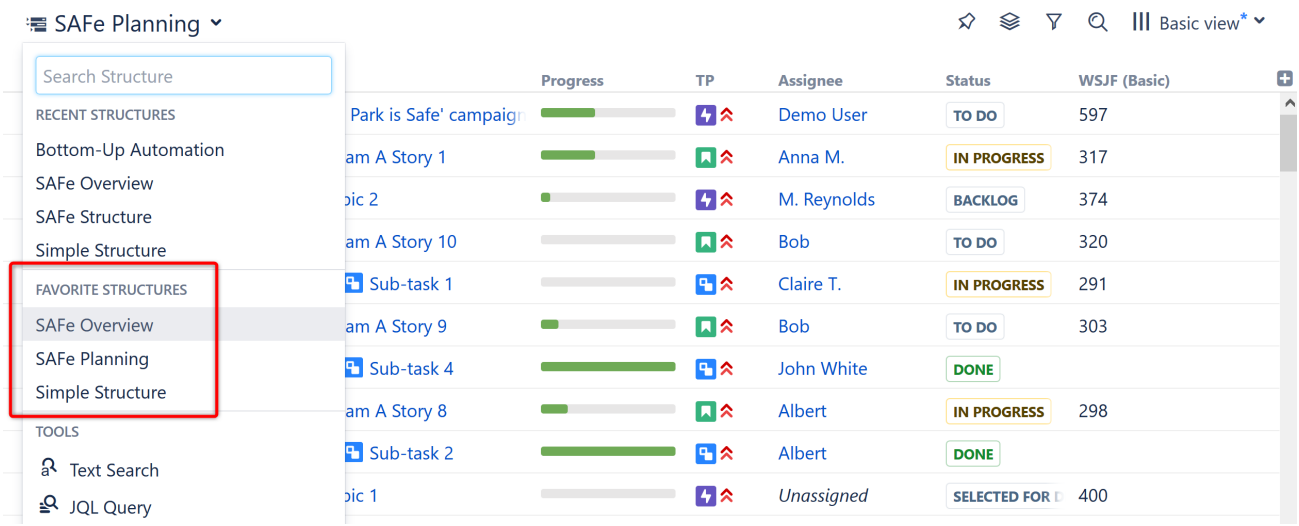
A default structure is displayed when no specific structure has been selected, typically the first time a user opens Structure. The default structure may also be displayed on Issue Pages(see page 88) and Project Pages(see page 90) when the **Auto-switch to default structure** option is selected.

Default structures can be set by:

- Jira administrators can change the system default structure(see page 482)
- Project managers can select a project-level default structure(see page 0) for any project that is enabled for Structure

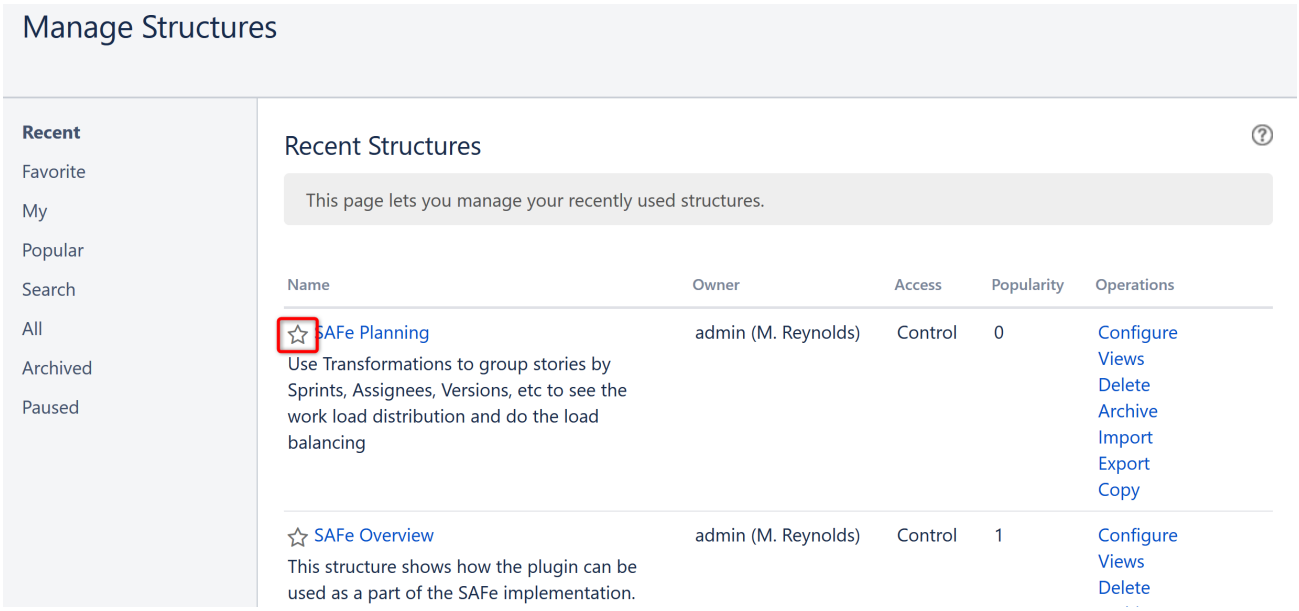
3.8.3 Favorite Structures

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.



3.8.3.1 Add/Remove Favorite Structures

To favorite a structure, open Manage Structures(see page 354) and locate the structure(see page 356) you want to favorite. Click the white star (☆) near the name of that structure. The star will then be shaded (★) to indicate that the structure was added to your favorite structures list.



To remove a structure from your favorites list, simply click the star again. To view all favorited structures, open Manage Structures(see page 354) and click the Favorite tab.

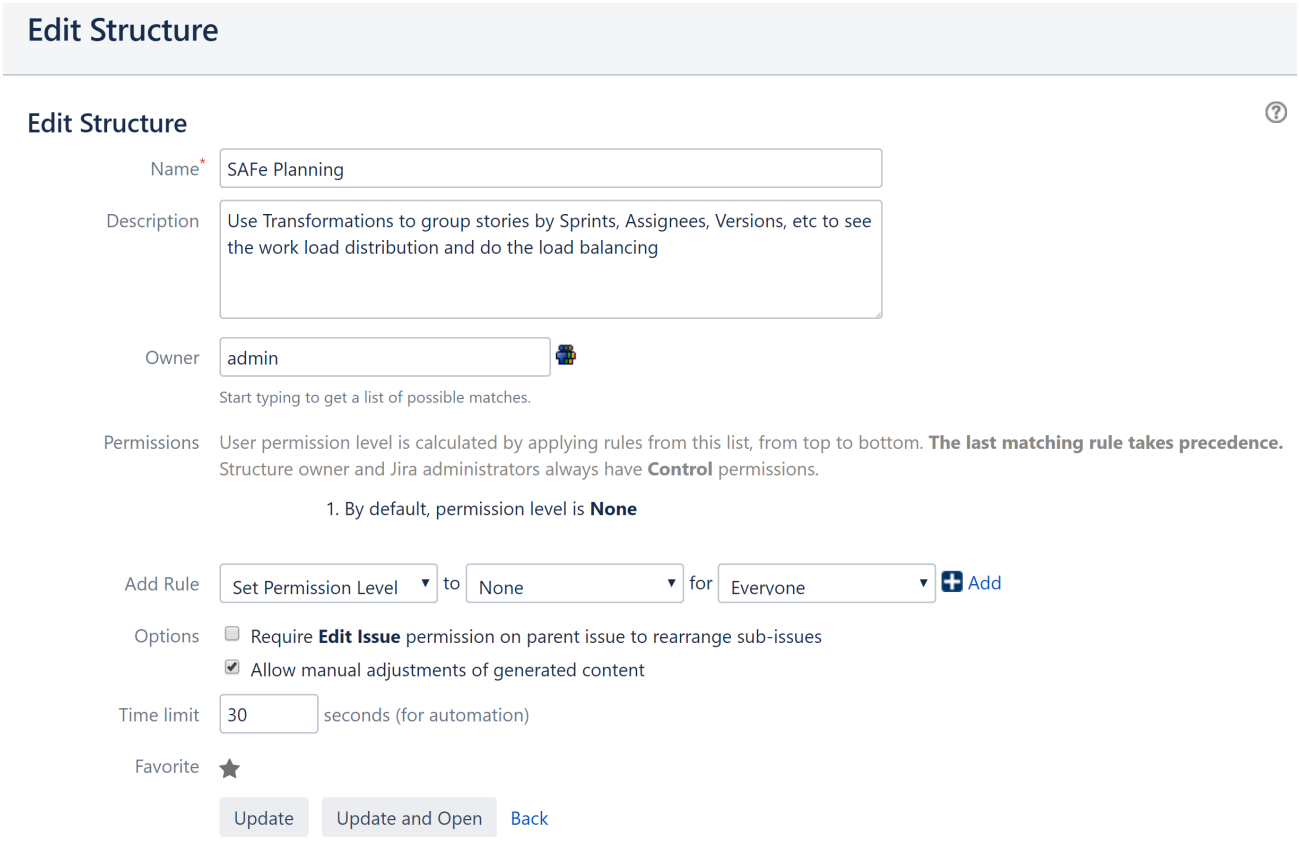
3.8.3.2 Structure Popularity

Structure **popularity** is the number of users who have marked this structure as a favorite. To view the most popular structures, open Manage Structures(see page 354) and click the **Popular** tab.

3.8.4 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(see page 362) for details.
- **Optional Settings:**
  - *Require Edit Issue permission on parent issue to rearrange sub-issues* - When set, users cannot move sub-issues unless they have permission to edit the parent issue as well. See Structure Permissions(see page 362) for details.
  - *Allow manual adjustments of generated content* - Enables or disables manual adjustments.(see page 175)
- **Time Limit** - Determines the maximum amount of time Automations(see page 122) can run before being paused(see page 172).
- **Favorite** - When selected (the star is filled in), the structure will appear in your Favorite Structures(see page 357) list.

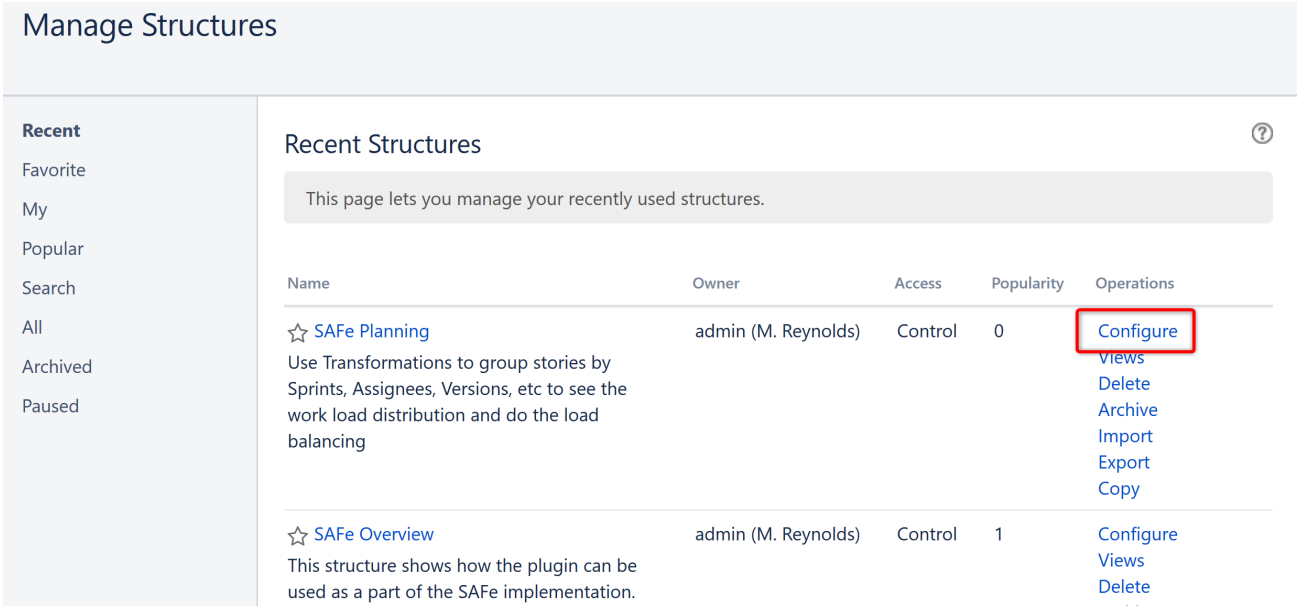


You can specify structure details when Creating New Structures(see page 365) and when Editing Structure Details(see page 360).

3.8.4.1 Editing Structure Details

The Edit Structure screen allows you edit details(see page 359) of a structure. To access the Edit Structure screen:

1. Open the **Structure** menu and select **Manage Structures**.
2. Locate the structure you need to change and click the **Configure** link in the **Operations** column.

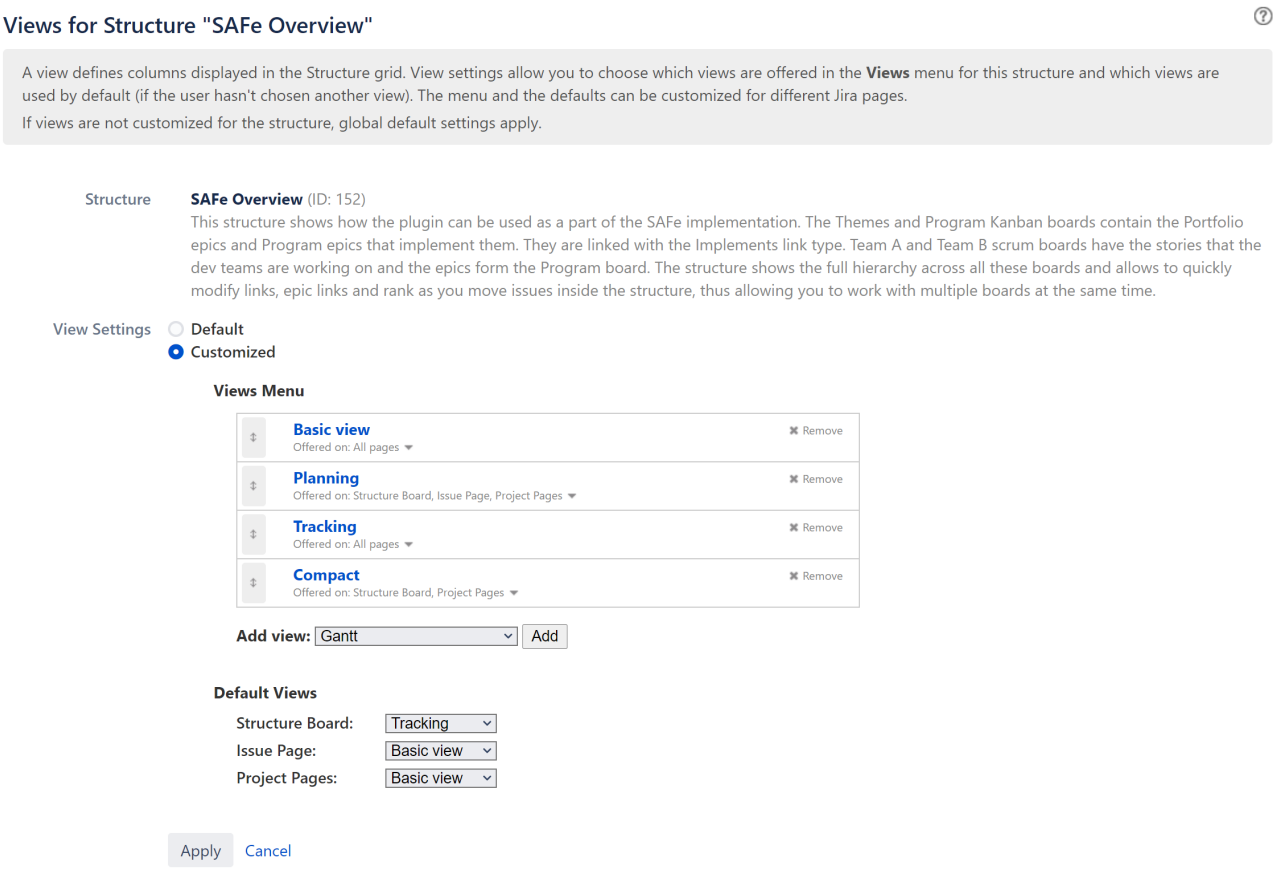


If you do not see a **Configure** link, you probably do not have Control permission for that structure.

See Structure Details(see page 359) for more information.

3.8.5 Customizing View Settings

A structure's view settings determine which views are offered to the users in the Views Menu(see page 337) and which view is used by default. To customize view settings for a structure, open the Manage Structures (see page 354)page, locate the structure and click the **Views** link.



You must have **Control** permissions for the structure to adjust its view settings.

3.8.5.1 Switching Between Default and Customized View Settings

Initially, each structure uses the default view settings, defined globally for all structures. To customize view settings for the structure, select the **Customized** radio button. The default settings are copied and you can adjust them to your needs.

To revert to the default view settings, select the **Default** radio button.

You can change the default global view settings if you are a Jira administrator – open the **Administration | Structure | Defaults** tab and click **Change** in the Default View Settings section.

3.8.5.2 Configuring Views Menu

The **Views Menu** section lets you select which views appear under the Associated Views section of the Views Menu(see page 337) for each type of Jira page(see page 83).

- To add a view to the menu, select the view in the **Add view** drop-down and click **Add**
- To remove a view from the menu, click the **Remove** button
- To change a view's position in the menu, drag the view by the drag handle at the left of the view bar

- To restrict a view's appearance in the menu to some specific pages, click the **Offered on:** line and select the pages where you'd like this view to be used

A view is **associated** with a structure if it is part of the Views Menu, as defined by the structure's view settings.

3.8.5.3 Configuring Default View

In the **Default Views** section, you can select which of the views included in the Views Menu section is the default view for a given Jira page (Structure Board, Structure Board with Issue Details, Issue Page and Project Page). Pick one view from those offered in the drop-down for each page.

If the **Views Menu** does not include any views for a specific page (for example, no views for Issue Page), you won't be able to configure the default view for that page.

Changes take effect when you press **Apply**.

3.8.6 Structure Permissions

Every structure has a list of permission rules, which define who is allowed to see, edit or configure the structure.

3.8.6.1 Access Levels

Each user has one of the following access levels to a structure:

None	The user does not see the structure at all and does not know that it exists.
View	The user can view the structure but cannot make changes.
Edit	The user can view the structure and can rearrange, add and remove issues from the structure. The user cannot, however, create or modify generators(see page 124).
Automate	The user has full edit access to the structure, including modifying generators(see page 124) and effectors(see page 154).
Control	The user can view, edit and configure the structure - including changing structure permission rules.

3.8.6.2 Default Access

By default, all users have **None** access level.

The structure's owner and Jira administrators always have **Control** access level.

Therefore, if you create a new structure and do not specify any permission rules, it will be a private structure that only you and Jira administrators will be able to see and modify.

3.8.6.3 Permission Rules

Users who have **Control** permission for a structure can define permission rules by Editing Structure Details(see page 360).

The Permission Rules list is an ordered list that's used to calculate the access level for a given user. Each rule assigns an **access level** to a specific **condition** (category of users).

The conditions are applied from top to bottom, and the **last matching rule has precedence** - so if a user fits multiple conditions, their access level will match the lowest-listed matching condition. For this reason, it is advisable to order permissions from least access (None) to most access (Control), as we've done above.

Setting Permissions

To set permissions, in the **Add Rule** section, select an access level and choose one of the following conditions for that access level:

Everyone	Matches any user, including anonymous (not logged in). This condition can be used to set a default permission for everyone.
Group(G)	Matches users that belong to the group G.
Project Role(R,P)	Matches users that have role R in project P.
User	Adds the permission for a specific Jira user.

To copy the permissions from another structure, click the **Set Permission Level** box and choose **Apply Permissions From**. This will let you apply the same permissions rules from any structure for which you have Control access level.

Permission Examples

The following are examples of how your permissions list might look:

- Everyone can view the structure, Jira administrators can edit, only the owner and admins can control:

  1. **View** for **Everyone**
  2. **Edit** for **jira-administrators** (Group)

- Any logged-in user can edit the structure, except for the users from the structure-noaccess group, who can't even view the structure. Project administrators are allowed to control the structure:

  1. **Edit** for **jira-users** (Group)
  2. **None** for **structure-noaccess** (Group)
  3. **Control** for **Administrators** of **SAFe Program** (Project Role)

- Incorrect configuration - in this example, everyone is given View access level:

  1. **Control** for **jira-developers** (Group)
  2. **Edit** for **jira-users** (Group)
  3. **View** for **Everyone**

  Although the configuration looks fine at first glance, remember that **the last matching rule has precedence**. So even if a user is part of the jira-developers or jira-users group, their access level will be set to View by the last rule.

Require Edit Issue permission on parent issue

When this option is selected on the Structure Details(see page 359) page, the user must have Edit Issue permission for a parent issue in order to adjust its sub-issues. In other words, direct sub-issues (or children issues) are treated as if they are part of the parent issue; therefore, adding sub-issues, removing sub-issues and rearranging sub-issues is actually changing the parent issue - for which the Edit Issue permission is required.

> The user must also have **Edit** access level to the structure to be able to make changes at all.

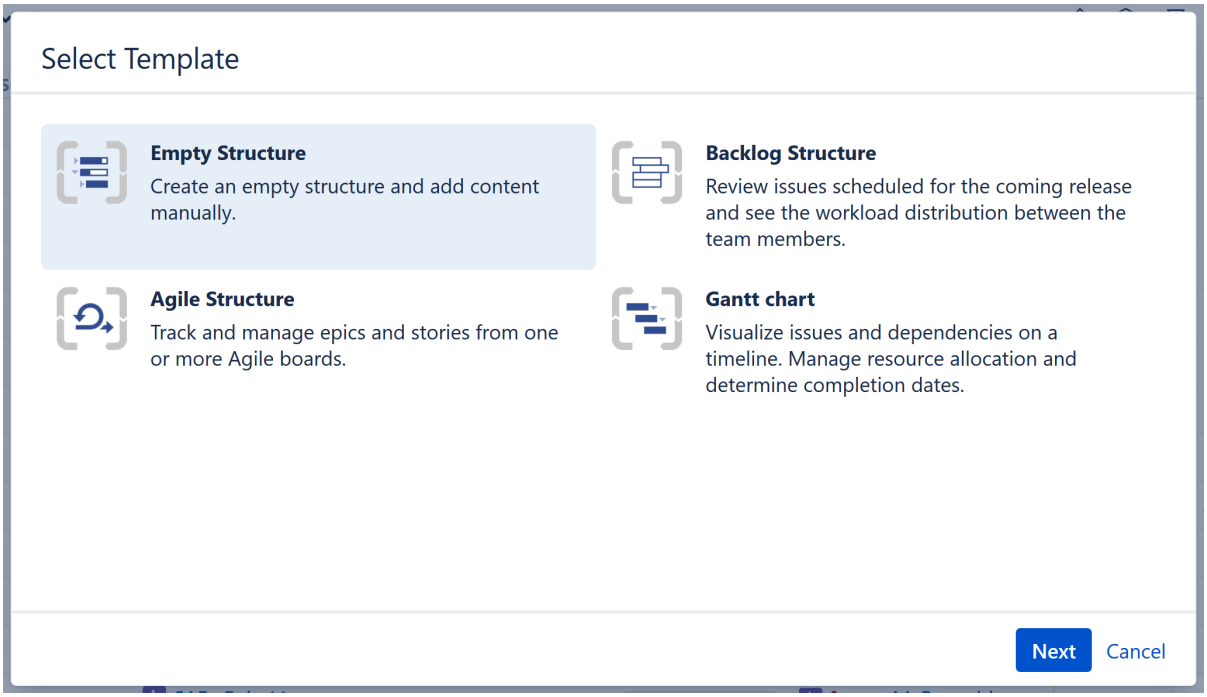
Note the following:

- Top-level issues do not have parent issues, and therefore are not affected by this flag: the user can add/ rearrange issues at the top level of the structure if they have Edit access level.
- The Edit Issue permission applies only to the direct parent issue. If issue A has sub-issue B, and B has sub-issue C, then to be able to move or remove C from the structure, the user needs Edit Issue permission on B - not on A.

> Structure maintains a cache of users permissions with regards to each structure. In most cases, the cache is recalculated automatically, but in some cases Structure may miss a change in a user's groups or roles. This could mean that the changed permissions for the user do not take effect until several minutes later (but only with regards to Structure Permissions(see page 362)).
>
> 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.

3.8.7 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(see page 354) 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 addition permissions later using the Structure Details(see page 359) page.

> When you create a new structure, you become the owner of the structure. Structure owner always has full access to the structure - see Structure Permissions(see page 362).

If you choose one of our templates, the template wizard will ask you a series of questions to help you add and organize issues to your specific needs. If you choose an empty structure, you can add (see page 94)and organize(see page 99) issues or apply automations(see page 122) once the structure is created.

> Only logged-in users who have access to Structure are allowed to create new structures. See Who Has Access to the Structure(see page 478) for more details.

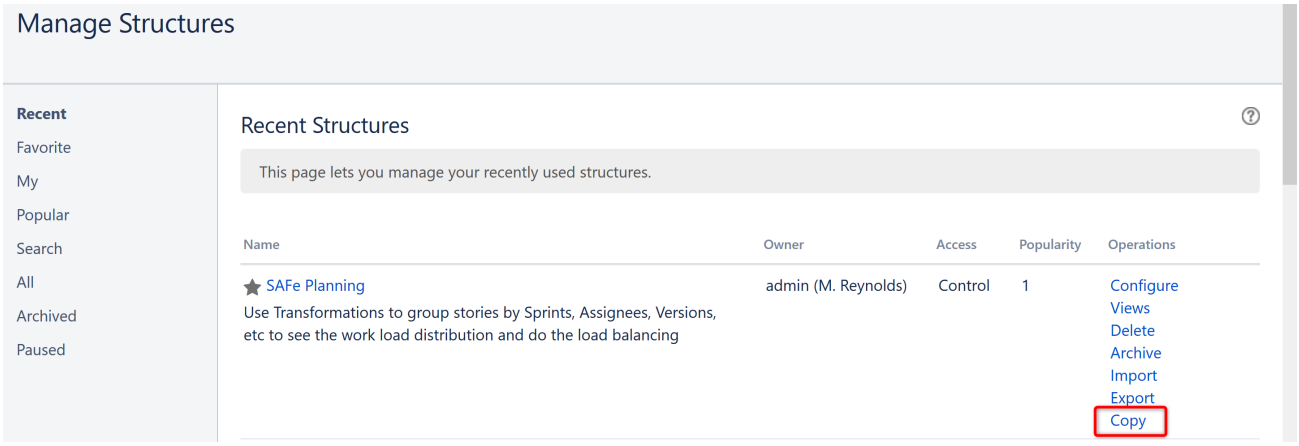
3.8.8 Copying a Structure

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(see page 103) to copy a part of the structure.

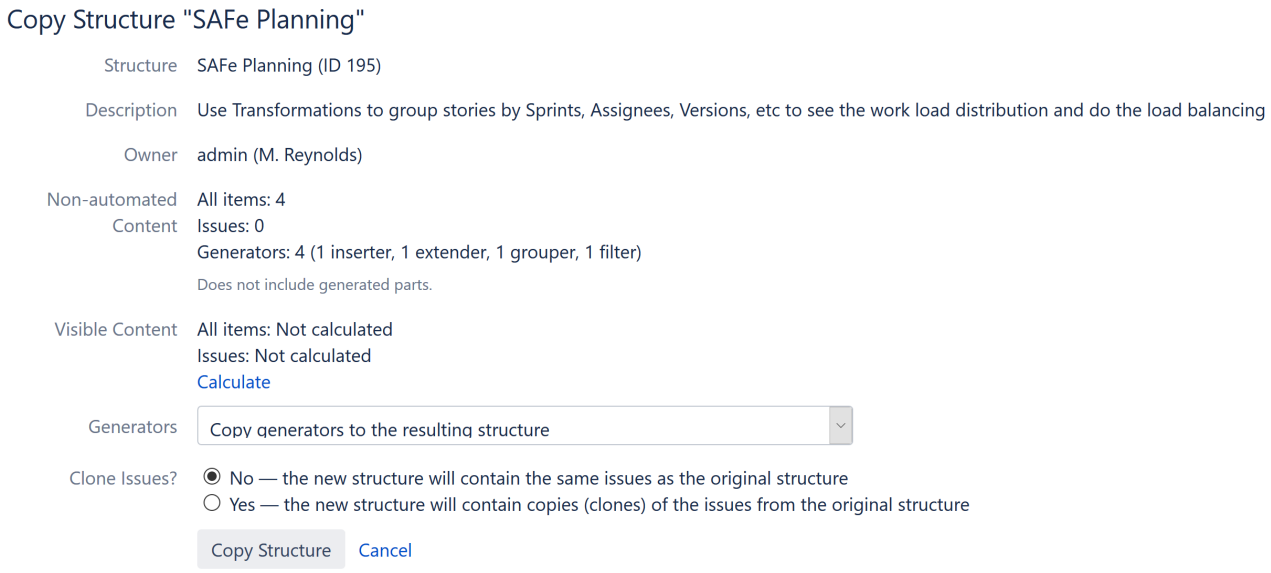
3.8.8.1 Create a Copy

To create a copy of a structure, open the Manage Structures(see page 354) page using the top navigation **Structure** menu. Find the structure you'd like to copy and click the **Copy** link in the Operations column.



If you don't see **Copy** in the Operations column, you probably do not have permissions (see page 479)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(see page 124), and synchronizers(see page 514) it contains. If the structure contains automation(see page 122), you can click the **Calculate** link in the **Visible Content** section to execute the generators and see the generated content statistics.



Select the following options for copying the structure:

1. **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.

2. **Choose if you'd like to copy** synchronizers(see page 514), if the structure has any. If you don't see a **Copy Synchronizers** option, either the structure does not have any synchronizers or you do not have permission(see page 522) to create synchronizers. See Copying Synchronizers(see page 538) for more details.
3. **Choose if you'd like to clone issues** (see Copy vs. Clone(see page 367)). If yes, enter the additional parameters for the cloning process(see page 368).

Once you've made you selections, press **Copy Structure** or **Start Cloning**.

3.8.8.2 The New Structure

The new structure is created with the following properties:

- Structure name is automatically set to "Copy of <*old structure name*> (<*date of copy*>)".
- Structure description is copied.
- View settings are copied.
- You become the owner of the copied structure.
- If you have **Control** access level to the original structure, permission rules are copied. Otherwise, permission rules for the new structure are empty (it is a private structure). To share the new structure, add permission rules(see page 362).

You can immediately edit the new structure's properties on the screen with the copy result.

3.8.8.3 Copy vs. Clone

When copying a structure, you can choose to have the new structure use the same issues as the original structure or clone the original issues (create copies). The following chart explains the differences:

	Copy Structure	**Copy Structure & Clone Issues**
Selected answer for **Clone Issues?**	**No**	**Yes**
New structure created?	Yes	Yes
New structure contains:	same Jira issues as the original structure	clones (copies) of the issues from original structure
Quick?	Yes	No, a background process is launched to do issue cloning

	Copy Structure	Copy Structure & Clone Issues
Permissions required:	**View** access to the original structure **Create Structure** permission	**View** access to the original structure **Create Structure** permission **Bulk Change** global Jira permission A number of project-level permissions

For details about configuring and running cloning, see Copying Structure and Cloning Issues(see page 368).

3.8.8.4 Copying Structure and Cloning Issues

When copying a structure(see page 365), 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:

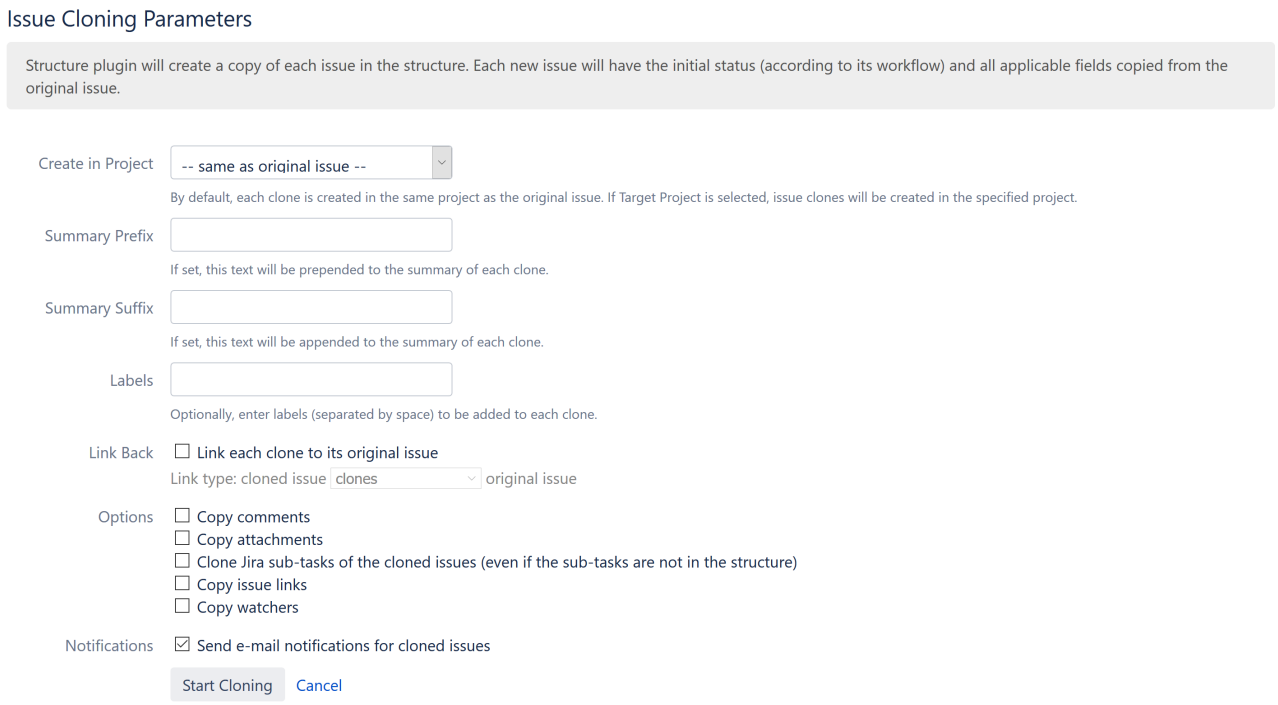
- If custom fields for the original issues are not available in the new project, the values of those custom fields are not copied.
- If field values of the original issues are not available in the new project, those values are removed. For example, this may happen to the **Components** field, or to fields 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.



Create in Project	Lets you specify a project for the new issues, different from the project the issues currently belong to. If not specified, every new issue is created in the same project as the original issue.
Summary Prefix **Summary Suffix**	Lets you modify the summary of the clones. If the resulting summary gets longer than the Jira limit (255 characters), it will be truncated.
Labels	Lets you add space-delimited labels to the cloned issues. (Already existing labels are preserved.)
Link Back	If specified, every new issue will be linked with its original issue.
Copy comments	If selected, all comments are also copied. If not selected, new issues will have no comments.
Copy attachments	If selected, attachments are copied (the actual files are copied on the Jira server).

Clone Jira sub-tasks of the cloned issues	If selected, when issues have sub-tasks in Jira which are not also in the structure, those sub-tasks will be cloned as well. For example, issue A-1 has sub-task A-2 in Jira. A-1 is in the structure being copied, but A-2 is not. With this option selected, A-2 will also be cloned. Otherwise, A-1 will be cloned without the sub-task.
Copy issue links	If selected, all issue links and remote issue links will be copied. If a link exists between two issues, and both are cloned, then the new link will be created between the clones. ⚠ If you use the **Link Back** option, then the links of the type selected for linking back to original issues will **not be copied**. Scrum board Epic-Story relationships are also copied when you select this option. The rule is the same as for issue links: • If you clone an epic together with its stories, the cloned stories will be added to the cloned epic. • If you clone the stories alone, the clones will be added to the original epic.
Copy watchers	If selected, the users watching an original issue will be added to the watcher list of the clone.
Notifications	If selected, an email may be issued for every created issue, depending on the Jira notification scheme for each issue's project.

Required Permissions

The following permissions are required to copy a structure with issue cloning:

- To be able to clone structure issues, you need **Bulk Change** global permission.
- Because the result of cloning is a new structure, you also need to be allowed to create new structures. (Configured by the Jira administrator - see Administrator's Guide(see page 479).)
- You need to have **Create Issue** permission in the projects where the clones are created. If you specify the **Create in Project** option, the issues will be created only in the specified project. Otherwise, clones are created in the same projects as their respective original issues.
- Users in the **Assignee** field of the original issues will have to have **Assignable User** permission in the target project – otherwise, cloned issues cannot be assigned to that user and will be assigned by default.
- If you don't have **Modify Reporter** permission, you won't be able to set the value of the **Reporter** field in the cloned issues. Instead of the original reporter, you will be the reporter of the issue clones.
- You need to have **Add Comments** permission to copy comments, **Link Issues** permission to copy issue links or use **Link Back**, **Create Attachments** permission to copy attachments, **Manage Watchers** permission to copy watcher lists, and **Edit Issue** permission to copy Epic-Story relationships.

Executing Bulk Cloning

When you press the **Start Cloning** button, a background process starts on the Jira server, which performs the following:

1. Copies the original structure's hierarchy and stores it in memory
2. Checks all necessary permissions required for cloning
3. Clones all issues
4. Creates a new structure and fills it with the cloned issues

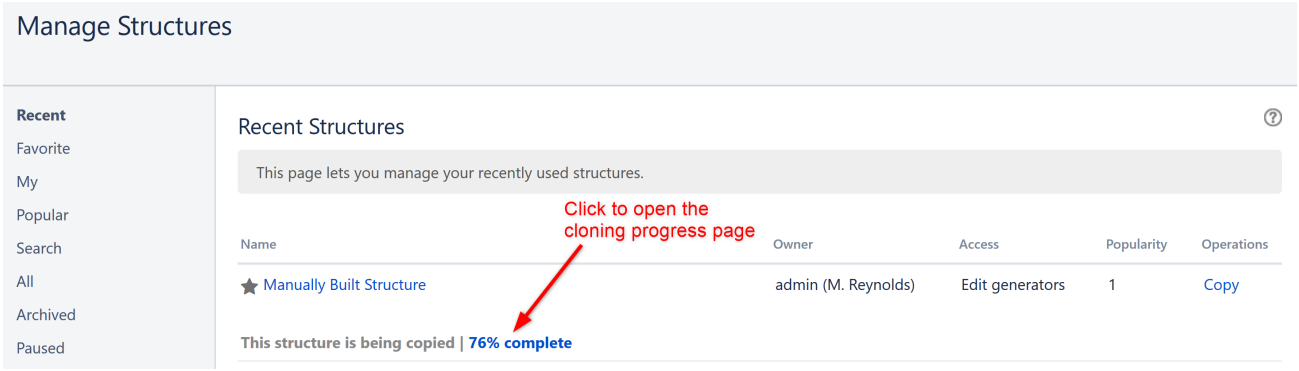
At step 2, the cloning process might discover some problems. If critical problems are discovered, an error message is shown and the process is aborted. If non-critical problems are discovered, warnings are shown and user input is required. The warnings may suggest that cloning may continue, but the resulting issues might not be exact copies. After your confirmation, the process continues.

As cloning proceeds, a progress bar is shown on the screen. When cloning is done, the Edit Structure page is opened for the resulting structure, so you can make any necessary adjustments to its name and permissions.

Cloning issues is potentially a long operation. Cloning a structure with tens of thousands of issues may take an hour or more. Cloning smaller structures should take considerably less time.

Checking Cloning Progress

When cloning has started, you can navigate away from the cloning progress page. To see the progress and get back to the progress screen, open the Manage Structures(see page 354) page and locate the original structure. It should show that the structure is being copied.



When cloning is completed, or if there are warnings or questions from the cloning process, the link will read "Waiting for input". Click the link to open the cloning progress page.

Cancelling Cloning

You can cancel the cloning process from the cloning progress page by pressing the **Cancel** link.

Issues that have already been created by the cloning process will be assembled into a special structure named "*[Cancelled Cloning Result]*". You can use Bulk Change(see page 120) to quickly delete the unwanted issues.

Cloning Queue

Cloning issues can place considerable load on a Jira server. To avoid overloading the server with cloning jobs, there is a limit to the number of cloning processes that can happen simultaneously. If this limit is exceeded, your cloning process will initially be in "waiting" state, pending for other cloning processes to finish.

3.8.9 Archiving a Structure

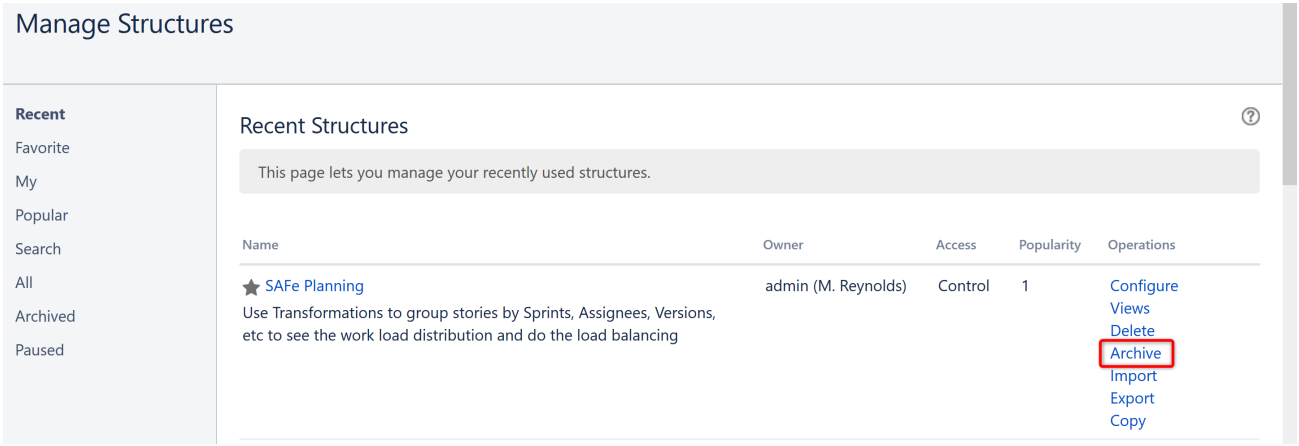
When you **Archive** a structure it becomes read-only and is hidden from search results and menus. The issues within the structure are not affected in any way. They remain in Jira and can still be part of another structure.

> *Read-only* means that users cannot add, remove or move items (issues, folders, memos, generators) in the archived structure.

3.8.9.1 Archive a Structure

To archive a structure:

1. Open the Manage Structures(see page 354) 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.



> 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.

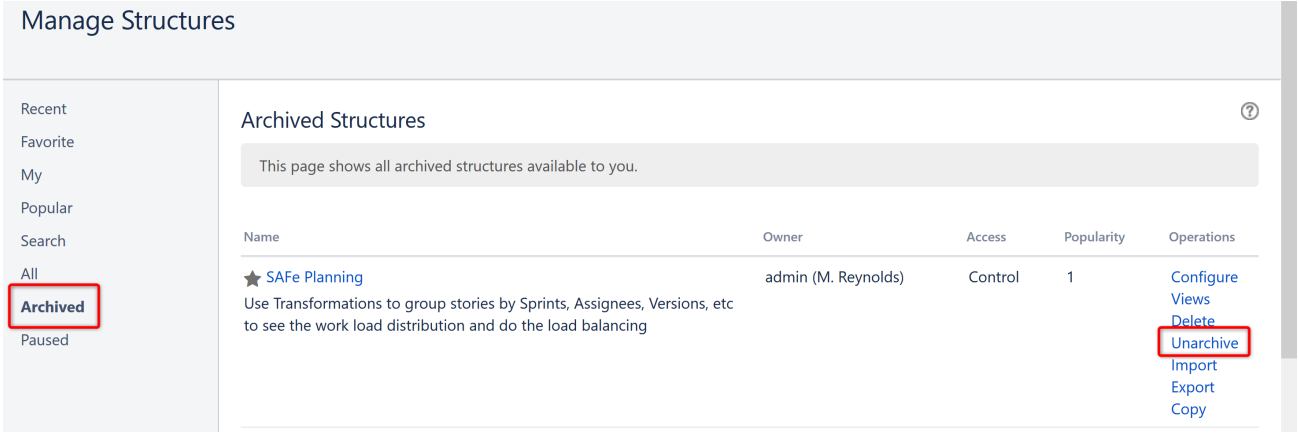
3.8.9.2 Unarchive a Structure

You can restore an archived structure to make it editable and visible in all menus.

To **unarchive** a structure:

1. Open the Manage Structures (see page 354)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.



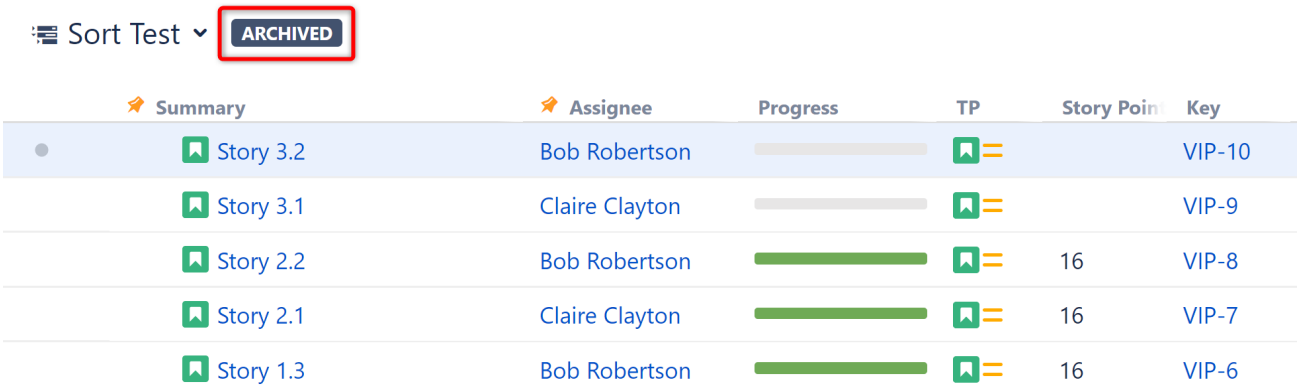
You need **Control** access level to be able to unarchive a structure.

3.8.9.3 Searching for Archived structures

Archived structures can be found on the following tabs on the **Manage Structures** page:

- **Archived** tab
- **Favorite** tab - if your favorites list contains any archived structures
- **Search** tab (Search by structure parameters) - you must check the **Show Archived** box
- **Search** tab (Search by the structure ID)

When an archived structure is opened, an **Archive** label appears next to the structure's name, so users can quickly identify it and know that it is read-only.



3.8.9.4 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(see page 516) an archived structure, if you have a special permission to control synchronizers.

3.8.10 Deleting a Structure

Deleting a structure cannot be undone. If there is a chance you may need the structure in the future, consider archiving(see page 372) it instead.

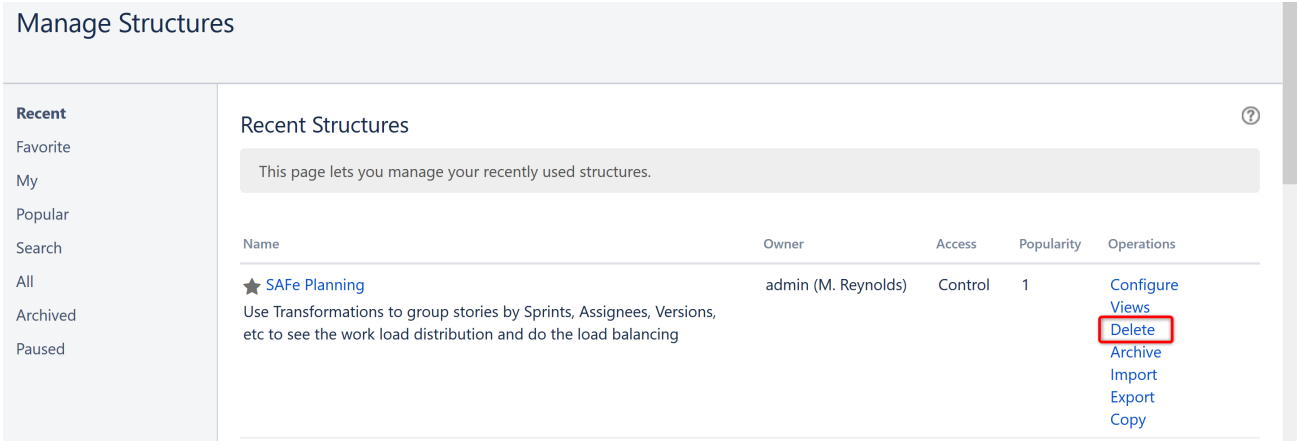
When you delete a structure, the following information is deleted:

- The hierarchical list of issues from the structure
- Structure details - name, description, permissions
- Automation rules for the structure
- Synchronizers installed into the structure

The issues within the structure are not affected in any way. They remain in Jira and can still be part of another structure.

To delete a structure:

1. Open the **Manage Structures** page using the top navigation **Structure** menu.
2. Find the structure you want to delete and click the **Delete** link in the Operations column.
3. Review the structure you are about to delete and confirm the operation. **Remember: this action cannot be undone!**



You need **Control** access level to be able to delete a structure. See Structure Permissions(see page 362) to learn more.

3.8.11 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.

3.8.11.1 Configuring Template Structures

Here are some suggestions about configuring template structures:

1. Clearly designate them as a template - for example, have "[Template]" marker as a part of the structure's name.
2. Give permissions to change the template structure only to those users who really need it. If needed, create another JIRA group for them (or ask JIRA administrator to do so).
3. Do not install any synchronizers on the template structure (unless you want the template to change, of course... which would be a quite unusual case).
4. Do not mark template issues as template in the issue summary. If you need to mark template issues somehow, use a label, which you will be able to remove from cloned templates via Bulk Change.

If you need to remove template issues from a JQL search, you can add to JQL: `AND NOT (issue in structure('template structure name'))`. See structure() JQL function(see page 266).

3.8.11.2 Creating Issues and a Structure from Template

Once you have a template structure, you can use Copy(see page 365) action from the **Manage Structures** page and turn on Clone Issues(see page 368) option. For details about configuring and running cloning operation, refer to Copying Structure and Cloning Issues(see page 368) article.

After you have created a new structure with new issues from template, you might want to:

- Rename the new structure and give it a meaningful name.
- Assign permissions for the new structure, if they are different from template structure permissions.
- Open the new structure to make sure it looks good.
- Do a Bulk Change(see page 120) on all issues - for example, to remove a template marker.

3.8.11.3 Template Projects

In the same manner, you can create a template project with template issues, and put them all into a template structure.

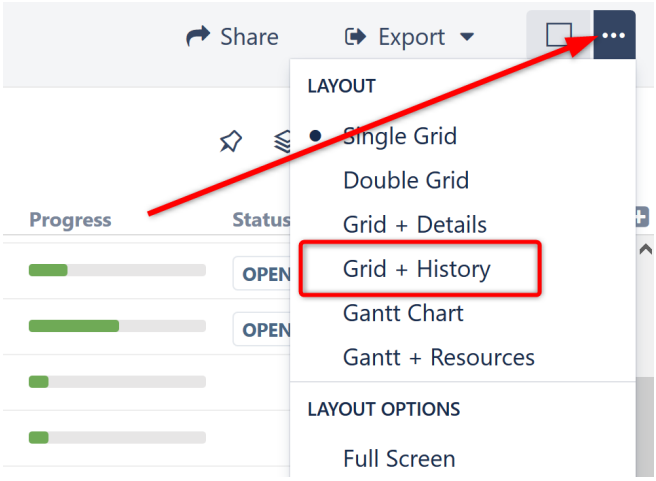
When you need to create a new project based on the template project, do the following:

1. Manually create an empty new project.
2. Create new structure and issues from template structure, as advised above. When configuring cloning parameters, specify the new project in the **Create in Project** parameter.

3.8.12 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 (see page 122)will not appear in the history. However, the addition, moving and removal of the generators themselves are recorded.

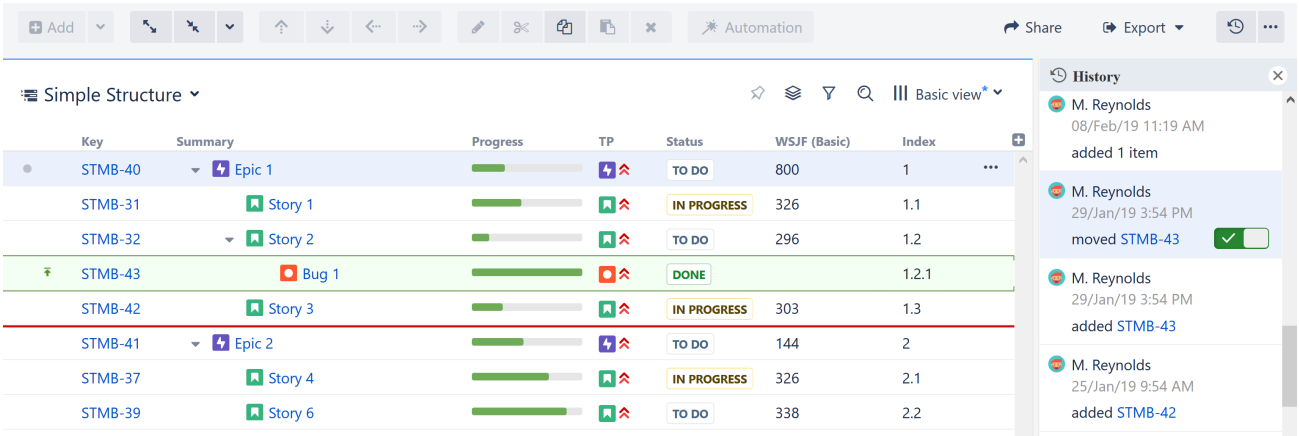
3.8.12.1 Reading History View

By default, the 20 most recent changes are loaded. To view additional changes, click the Show More button at the bottom of the list.

For each change, the following information is shown:

- The avatar and the name of the user who has made the change.
  - If the change was been made by a synchronizer, the synchronizer's name is shown. User avatar displays the user account that the synchronizer was running under.
- The nature of the change – how many issues were affected, were they added, removed or moved.
- The date and time when the change was made.

When you click a particular change, the main panel of the widget shows the structure as it was when that change was made. The affected issues are highlighted, and the structure expands and scrolls as needed to bring them into view.



Moved issues are shown in their new position by default, and their original position is marked by a red horizontal line. If you prefer to view moved issues in their original position, click the toggle button next to the entry in the history panel.

If issues were removed, they are shown in their position before the removal.

Use the **Ctrl+]** and **Ctrl+[** keyboard shortcuts to navigate to an earlier or later change.

3.8.12.2 Limitations of the History View

- History only tracks changes to the structure, not changes to Jira fields. Therefore, the values displayed in each column represents current Jira field values, **not** the values issues had when the structure change was made.
- History does not display changes that occurred through Automation(see page 122) - though it does display changes to the generators themselves.
- You cannot edit issues, create new issues or change structure when viewing history.
- The history cannot be modified. (The administrator can clear the entire Structure history.)

3.8.12.3 Printing a Previous Structure Version

You can create a printable version(see page 380) of the structure as it appeared following a given change. To do so, select the relevant change and click **Export | Printable Page**.

Please keep in mind that the printed structure will have the same limitations as the History View (Jira fields will display current values, etc.).

3.8.12.4 Exporting a Previous Structure Version to Excel

You can also export the structure to Excel(see page 378). To do so, select the relevant change and click **Export | Export to Excel**.

The XLS file will contain the structure as it appeared after the selected change was applied, with the same limitations as the History View (Jira fields will display current values, etc.).

3.8.13 Exporting Structures

There are a several options for exporting structures.

- Exporting Structures to PDF(see page 377)
- Exporting Structure to XLS (Excel)(see page 378)
- Printing Structure(see page 380)

3.8.13.1 Exporting Structures to PDF

You can use one of the following methods to export a structure (or part of a structure) to PDF.

The built-in way: Print the structure (see page 380)and use your browser's built-in Print to PDF feature to turn it into a PDF.

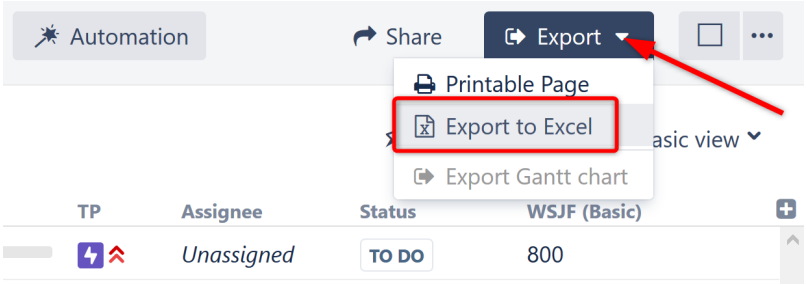
The more convenient way: Use one of these partner add-ons to customize and even automate the export process. PDF export is not our specialty - it is for these amazing apps!

- Better PDF Exporter for Jira(see page 405)
- Xporter(see page 421)

3.8.13.2 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**.

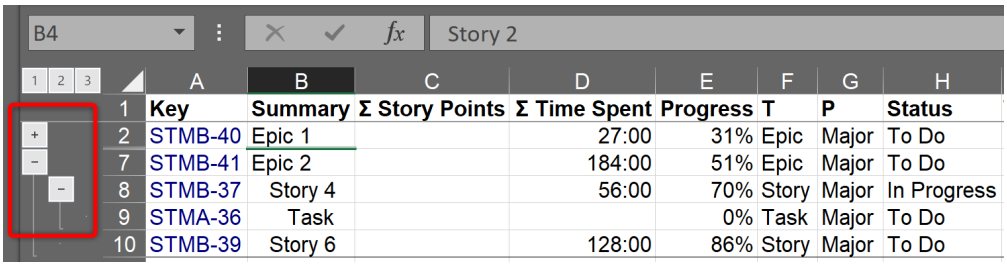


The browser will download a new XLS file, which you can save or open. The XLS file will:

- 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 (see page 333)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.



	A	B	C	D	E	F	G	H
1	Key	Summary	Σ Story Points	Σ Time Spent	Progress	T	P	Status
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(see page 380).

Columns

The columns are formatted in the best way suitable for a spreadsheet.

Column Type	Notes
Issue Key	The cell with an issue key is a link to the actual issue.
Summary	Cells in the Summary column have indentation just like in Structure. Note that if you change the format of a cell, you might lose the indentation level.
Progress	Progress field contains a fractional number from 0 to 1, formatted as a percent value.
Description, Environment and large text fields	The text might not fit in the column. You can increase the column size or use the **Format Cells | Alignment | Wrap Text** option in Excel to have a large text take up more than one line, increasing the row height. Note that a cell might not accommodate a very large text, and you might only see the first part of it.
Dates	Date values are displayed in your local date format.
Estimates, Time Worked	Duration fields contain actual numbers (fractional number of days), which you can sum or otherwise process. The display format is **HH:MM**, where HH is the number of hours and MM is the number of minutes. So an estimation of 5 days will be displayed as **40:00** (if you have 8-hour work days).
Standard custom fields	Standard custom fields are rendered according to their type.
Custom fields from other add-ons	Custom fields from other add-ons are displayed as they are rendered.

Compatibility

The exported file is compatible with Microsoft Excel 2003+.

Technical Limitations

The XLS format allows a maximum of 65536 rows in a spreadsheet. If your structure contains more items than this, use filtering to hide some of the issues.

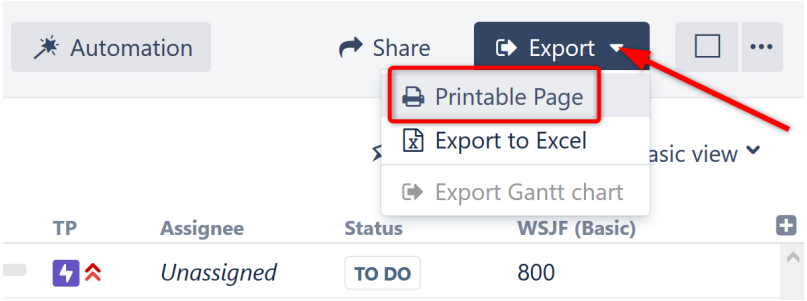
If your structure's hierarchy is more than 15 levels deep, any items deeper than level 15 will be indented to the 15th level in the Summary column of the XLS file.

Note for Add-on Developers

If your plugin provides a new custom field type, please ensure that the field is displayed with the best compatibility with the other plugins, including Structure. In your column view velocity template, check for `$displayParams.textOnly` and/or `$displayParams.excel_view` and/or `$displayParams.nolink` – all those parameters will be set to `true` by Structure and may also be used by other plugins. See `CommonVelocityKeys.java` and Jira sources for examples

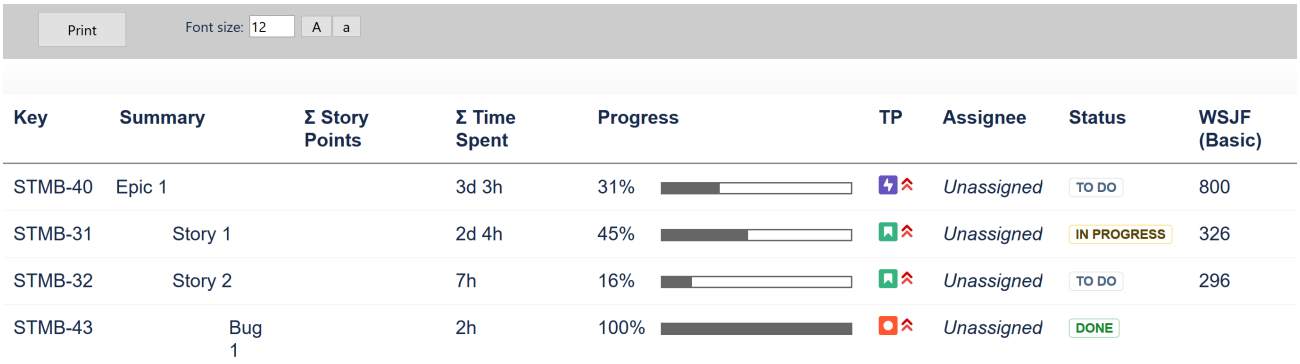
3.8.13.3 Printing Structure

To print the current structure, click the **Export** button on the structure toolbar and select **Printable Page**.



The structure will open in a separate browser window or tab, with the following properties:

- It copies the current appearance of the structure, so if some sub-issues are hidden, you will not see them on the printable page.
- The columns displayed on the printable page will be the same as in the structure; however, column widths will be set by the browser. Note: if horizontal scrolling (see page 333)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%		*Unassigned*	TO DO	800
STMB-31	Story 1		2d 4h	45%		*Unassigned*	IN PROGRESS	326
STMB-32	Story 2		7h	16%		*Unassigned*	TO DO	296
STMB-43	Bug 1		2h	100%		*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.

3.8.14 Real-Time Collaboration

Structure is a real-time collaboration tool. The hierarchy displayed in a structure is kept up-to-date with the Jira server, so:

- If someone else changes the structure on the server, you will see the web page update within several seconds.
- If someone edits an issue or otherwise changes it, the field values displayed within a structure will also be changed.
- If any part of a structure is built using Automation(see page 122), the inclusion and placement of issues within the structure will automatically update based on relevant changes in Jira.

Items that have been added, removed, moved or changed are highlighted for a second with a flashing yellow background.

Structure keeps data up-to-date by polling the server with short requests every few seconds when the application is in use. If Structure detects that the browser is inactive or that the user has switched to a different browser tab, it will reduce polling frequency or possibly stop polling altogether to conserve network traffic.

Polling should resume as soon as Structure detects activity, but if you ever feel that data may be out of sync, simply move the mouse or select a new issue.

3.8.15 Structure Activity Stream

JIRA's **Activity Stream** dashboard gadget lets you see recent activity in JIRA and other connected systems. The activity stream can be filtered (for example, by project) to show you only the changes that concern you or your team. In addition, **Activity** tab on the issue page displays recent activity that has affected the viewed issue.

With the Structure plugin installed, Activity Stream gadget may be configured to include changes made to structures. The activity stream on the issue page automatically includes all changes to all structures that affect the position of the viewed issue.

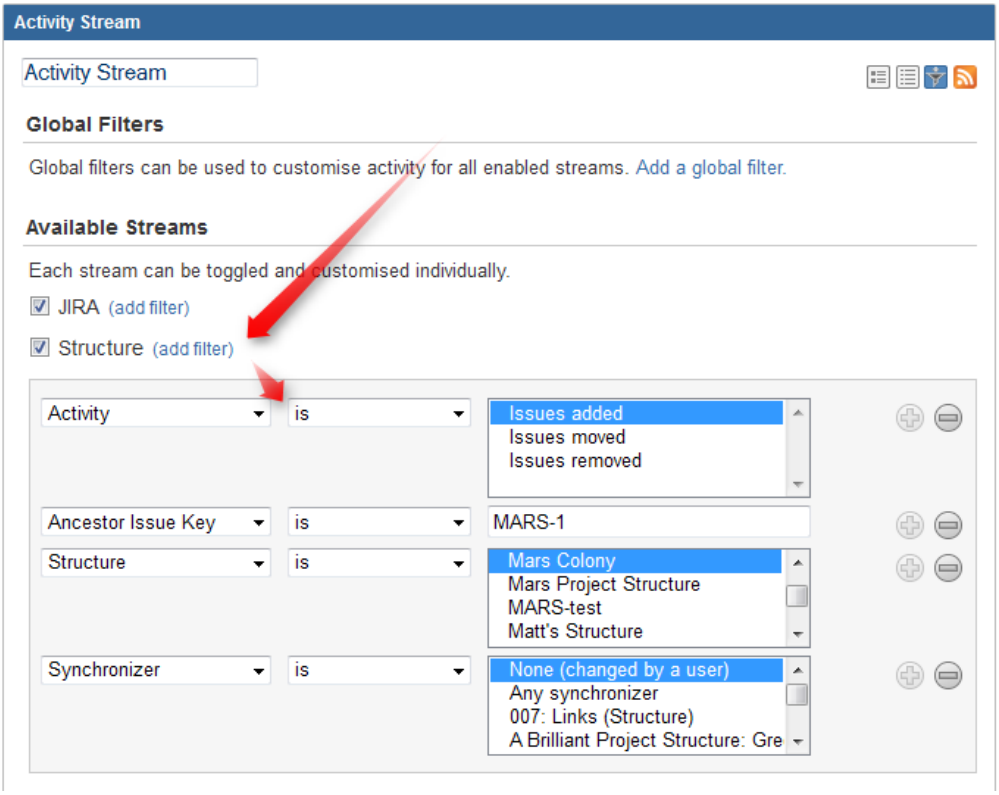
To activate the Structure stream, select the Structure option in the Available Streams section of the Activity Stream gadget configuration.

3.8.15.1 Available Filters

The following filters are available for the Structure activity stream:

- **Structure**
  Use it to see changes only in a specific structure or structures, or to exclude specific structures from the stream. If this filter is not used, changes to all structures are shown.
- **Ancestor Issue Key**
  This filter can be used together with the **Structure** filter if you are interested in changes within a specific part of a specific structure, located "under" the specified issue (if the changed issue is not located under the specified issue, the change will not be shown). You can enter several issue keys separated by spaces.
- **Synchronizer**
  You can include or exclude changes made by a synchronizer (either by any synchronizer or by specific synchronizers). Since synchronizers might make a lot of changes, this might be useful to filter out their "noise". Vice versa, you could verify that a synchronizer works as expected with an activity stream and this filter.
- **Activity**
  All changes to a structure fall into three categories: adding issues to structure, removing issues from structure and moving issues within structure. This filter lets you include or exclude the particular types of changes.

All Global Filters are supported by Structure Stream as well – you can filter structure changes by **Project, Issue Key, Update Date** and **Username**.



3.8.15.2 Reading Activity Stream

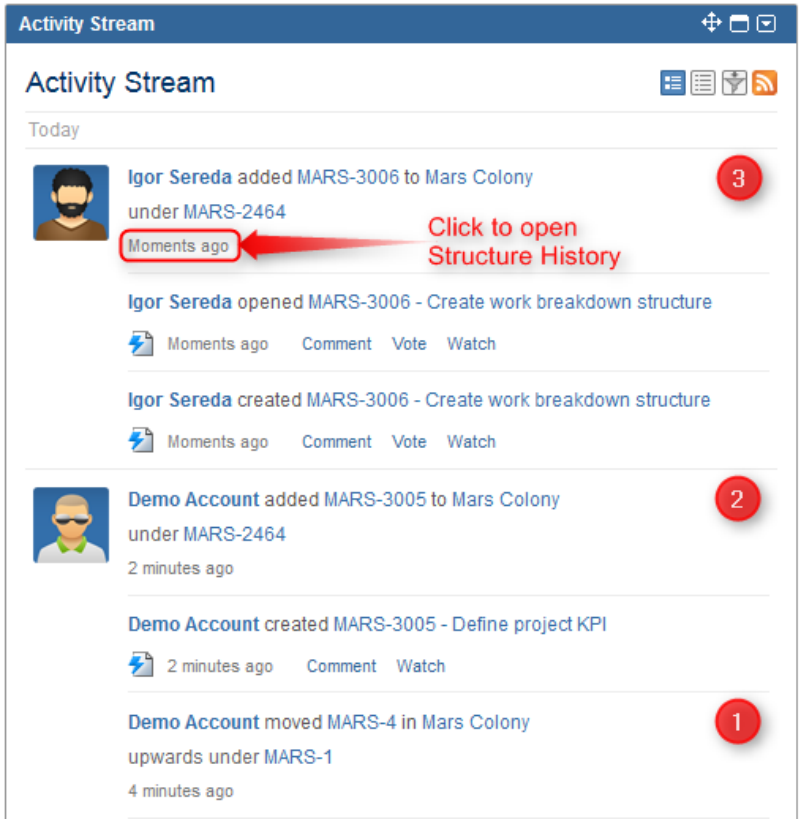
Changes in the Structure activity stream are ordered chronologically, newest first. For each change a short summary is displayed, containing:

- the full name of the user who made the change;
- for changes made by a synchronizer, the name of the synchronizer;
- the number of affected issues, and whether they were they added, removed or moved;
- if **Project** filter is used, the number of affected issues in each of the selected projects;
- if **Issue Key** filter is used, the affected issues among those selected in the filter;
- the name of the changed structure.

When viewing activity stream in the Full View, the following is also shown:

- the parent path of the affected issues;
- the original and the new parent path for the moved issues;
- if the issues were moved within the same parent issue, the direction of the move (upwards / downwards);
- when the change was made.

Parent Path is a sequence of issue keys: first, a top-level issue, then its sub-issue, then sub-sub-issue, and so on until the parent of the affected issue is displayed. Hover mouse over an issue key to view the issue's summary, or click it to go to that issue.



On this screenshot, items 1, 2 and 3 are Structure activities.

In the Full View, click on the time of the change to open that change on the Structure Board in the History View(see page 375).

3.8.15.3 Activity Streams Performance

Structure's activity stream is optimized to quickly provide data for the most common activity requests from Dashboard, Issue Activity, User Activity and Project Activity page.

It is possible however, if you use a complex search query on a JIRA instance with large history of structure changes, that querying database will take longer time than Activity Streams allows and you will not see any results. (There should be a message that "one of the activity streams providers took long time to provide an answer".)

If that is the case, try to reduce the amount of conditions you are using or contact support for help.

3.9 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.

- Keyboard Shortcuts (PC)(see page 385)
- Keyboard Shortcuts (Mac)(see page 389)
- Quick Action Lookup(see page 393)

3.9.1 Keyboard Shortcuts (PC)

3.9.1.1 Navigation

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

3.9.1.2 Structure Views

Action	Shortcut
Switch View	vv
Return to Previous View	vvv
Save View	vs

Action	Shortcut
Save View As	`vss`
Revert Changes to View	`vr`

3.9.1.3 Searching & Adding to Structure

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

3.9.1.4 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+↓`

3.9.1.5 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+↓

3.9.1.6 Changing Issues

Action	Shortcut
Edit Field	*Double-Click*
Edit Summary	`Tab` *or* `F2`
Finish & Save	`Enter` *or* `Ctrl+Enter`

Action	Shortcut
Cancel Field Changes	`Esc`
Edit Next Field	`Tab` *or* `Ctrl+Alt+→`
Edit Previous Field	`Shift+Tab` *or* `Ctrl+Alt+←`
Edit Next Issue	`Ctrl+Alt+↓`
Edit Previous Issue	`Ctrl+Alt+↑`

3.9.1.7 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+↑`)

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

Clear Selection	`Escape`

3.9.1.8 Advanced

Action	Shortcut
Hide/Show Resolved	`rr`
Cut (Prepare to Move)	`Ctrl+x`
Paste (Move)	`Ctrl+v`
Paste Sub-Issue (Move)	`Ctrl+Shift+v`
Fix/Unfix View on Issue	`Ctrl+.`
Switch Panel	\
View Full-Size Image(see page 314)	`ii`
Show/Hide Issue Details without Switching Panel	`Shift+o`
Show Automation	~

3.9.2 Keyboard Shortcuts (Mac)

3.9.2.1 Navigation

Action	Shortcut
Select Issue	*Left-Click*
Show/Hide Issue Details	o
Previous Issue	↑ *or* k
Next Issue	↓ *or* j

Action	Shortcut
Expand Sub-Issues	→
Collapse Sub-Issues	←
For Large Structure	↟ ↡ ↖ ↘
Add Column	tt
Expand All	++
Collapse All	–
Change Structure	xx

3.9.2.2 Structure Views

Action	Shortcut
Switch View	vv
Return to Previous View	vvv
Save View	vs
Save View As	vss
Revert Changes to View	vr

3.9.2.3 Standard JIRA Actions

Action	Shortcut
Operations Dialog	.

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

3.9.2.4 Changing Structure

Action	Shortcut
Move Up	^↑
Move Down	^↓
Indent	^→
Outdent	^←
Drag and Drop	⇧Drag
New Issue	⏎
New Sub-Issue	⇧⏎
Remove from Structure	⌦

3.9.2.5 Changing Issues

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

3.9.2.6 Selecting Issues

Action	Shortcut
Toggle Selection	`space`
Select All	⌘a
Select All Sub-Issues	⇧→
Deselect All Sub-Issues	⇧←
Expand Selection Down (Up)	⇧↓ (⇧↑)

Action	Shortcut
Bulk Selection	⇧↟ ⇧↡ ⇧↖ ⇧↘
Cancel Selection	esc

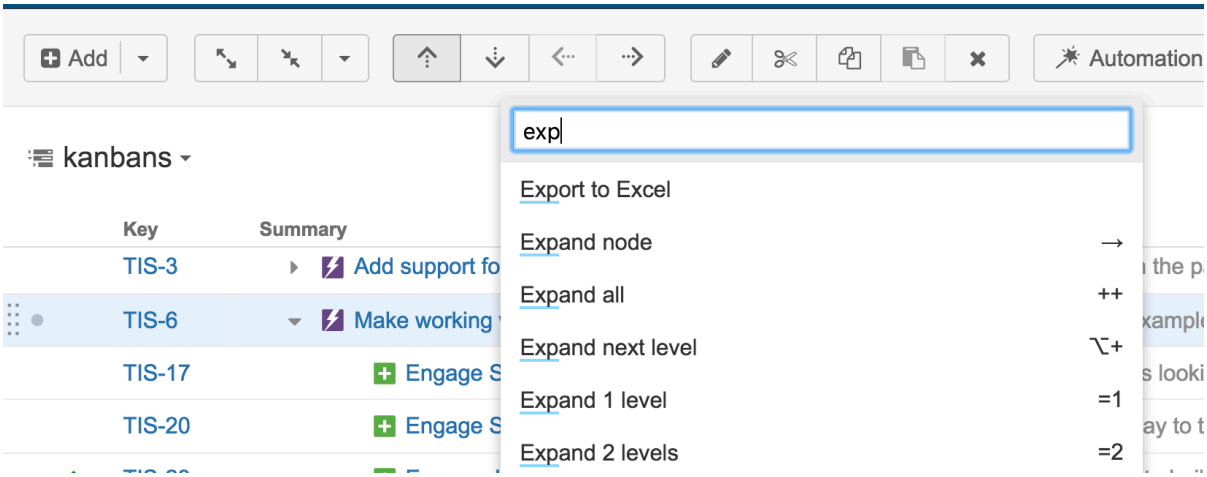
3.9.2.7 Advanced

Action	Shortcut
Hide/Show Resolved	rr
Cut (Prepare to Move)	⌘x
Paste (Move)	⌘v
Paste Sub-Issue (Move)	⌘⇧v
Fix/Unfix View on Issue	^.
View Full-Size Image(see page 314)	ii
Show/Hide Issue Details without Switching Panel	⇧o

3.9.3 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.

3.10 Structure Gadget

Structure gadgets can be added to the Jira dashboard or Confluence pages.

See the following articles to learn more.

3.10.1 Dashboard Gadget

You can view and edit structures directly from your Jira dashboard with the Structure dashboard gadget.

3.10.1.1 Adding Structure Gadget to Dashboard

Structure gadget can be added like any other dashboard gadget:

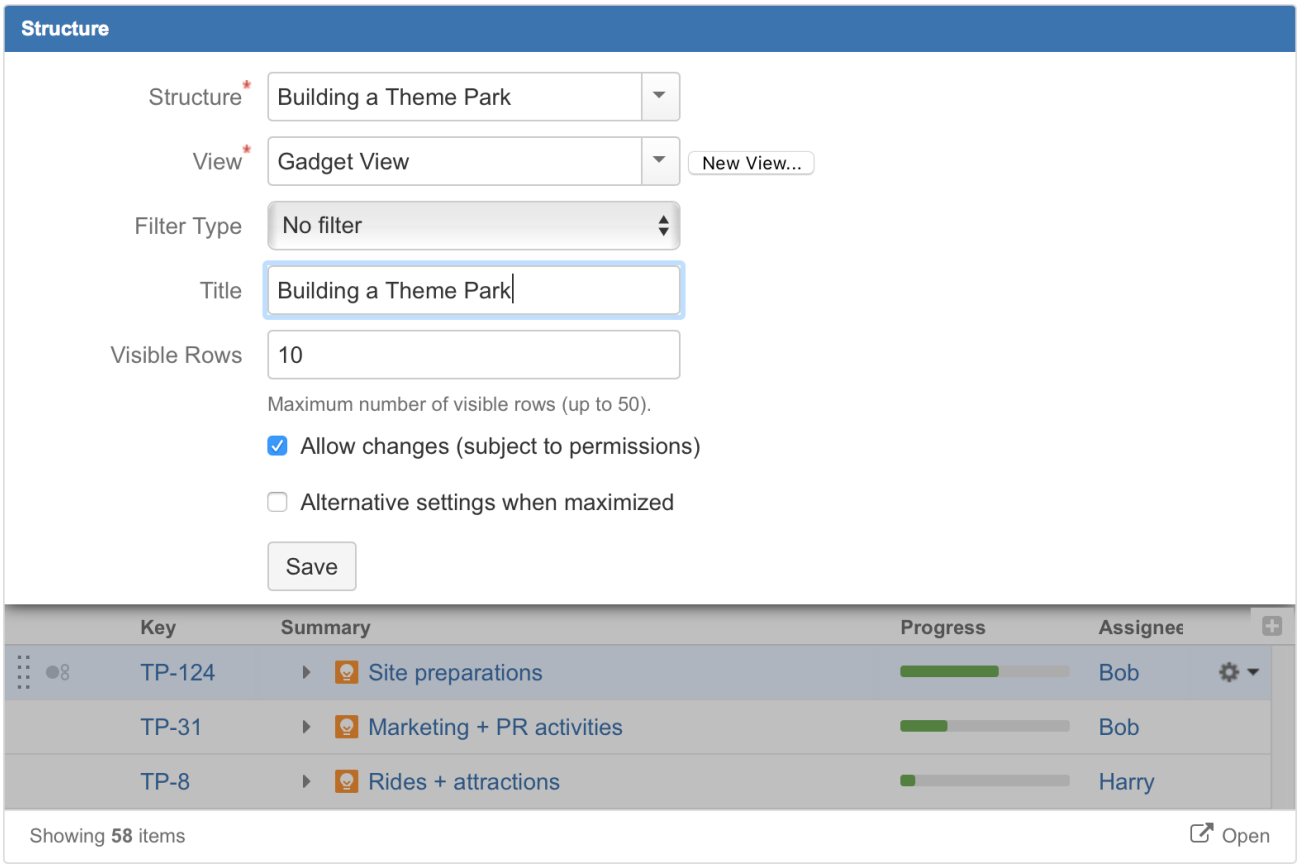
1. Click the **Add Gadget** button in the top right corner of the dashboard
2. Find "Structure" in the list of available gadgets (Hint: if you don't see Structure, click the **Load all gadgets** link)
3. Click **Add It Now**

In order to add a gadget to an existing dashboard, you must have change permissions. If you do not have change permission, you can try to create a copy of the dashboard using **Tools | Copy Dashboard**.

You can add several gadgets showing different structures on the same dashboard.

3.10.1.2 Configuring the Gadget

When you first add a gadget to dashboard, the gadget configuration panel appears with a dimmed preview of the gadget below. (The same panel is shown when you use the **Edit** command from the gadget header drop-down.)



To configure the gadget:

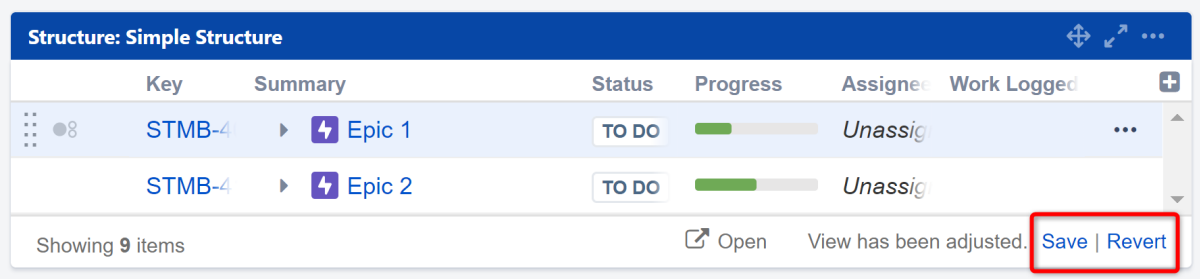
1. Select a **Structure**. Click arrow down in the Structure selector to view recently used and favorite structures, or start typing the structure name and let the drop-down suggest matching structures.
2. Select a **View**. Click arrow down to choose from views associated with the selected structure, start typing and let Structure suggest matching views, or create a new view. The selected view(see page 336) determines which columns the gadget displays. (You will be able to adjust the view later.)
3. Optionally, configure a **Filter**. The displayed structure will be filtered in the same way as in the Structure Board – see Filter(see page 183). You can choose between a text filter, JQL, S-JQL, or a saved JQL filter – see Search(see page 180).
4. Optionally, define the **Title** for this gadget. By default, it is the name of the selected structure.
5. Decide how large the gadget is allowed to be and specify the number of **Visible Rows**. If there are fewer visible rows, the gadget shrinks; if more, a vertical scroll bar appears. Pick any number between 2 and 50.
6. Decide whether dashboard viewers can make changes to the structure or issues (subject to the user's permissions) by selecting or deselecting the **Allow Changes** checkbox.
7. Optionally, if you would like the gadget to have a different **View** and **Visible Rows** settings when maximized, select the **Alternative settings when maximized** checkbox. This allows you to see more information for the same structure when the gadget window is maximized.
8. Click **Save**.

Deselect **Allow Changes** to protect the structure from accidental changes, such as changes caused by drag-and-drop or hitting the Delete key.

3.10.1.3 Customizing Gadget View

Once you have created your gadget, you can customize by adding, removing or rearranging columns – see Customizing Columns(see page 331).

Once you've made changes to the gadget's view, a new indicator will appear at the bottom of the gadget, along with options to save those changes or revert back to the view's original setting.



Save

Click the **Save** link will apply all the changes you have made to the existing view. The view will be updated everywhere it is in use, in the gadget, on Structure Board, etc. It will also impact all other users with access to that view.

> You must have Update permission(see page 346) for the current view to save changes.

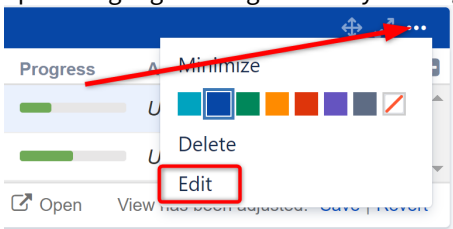
Revert

Click the **Revert** link to remove your adjustments and return to the original view as it is stored on the server.

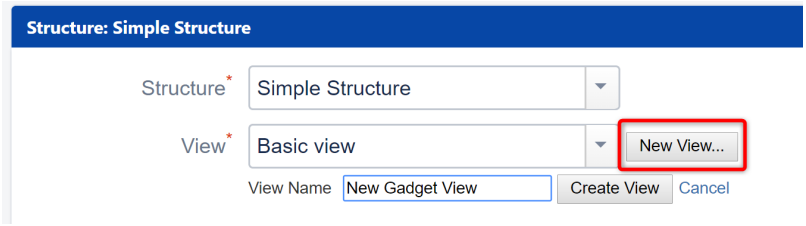
Save As

There is no Save As option on the dashboard gadget; however, it can be accomplished using the following method:

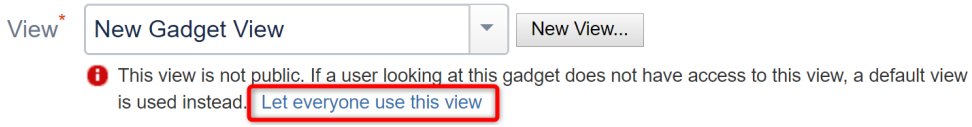
1. Adjust the gadget's view by adding, removing or rearranging columns – see Customizing Columns(see page 331) for details.
2. Open the gadget configuration by clicking **Edit** in the gadget header drop-down menu.



3. Click the **New View** button, located beside the view selector. An additional form appears – enter the new view name and click **Create View**.



4. If this gadget is going to be visible to other users, make sure they have access to the view you've created by clicking **Let everyone use this view**.



Use this method if you don't have **Update** access to the original view or want to continue using the original view elsewhere.

If the user viewing the gadget does not have **Use** permission on the configured view, the gadget will show a default view with only Issue Key and Summary as columns.

3.10.1.4 Using the Gadget

When viewing a structure in a gadget, some of Structure's features will be limited or unavailable:

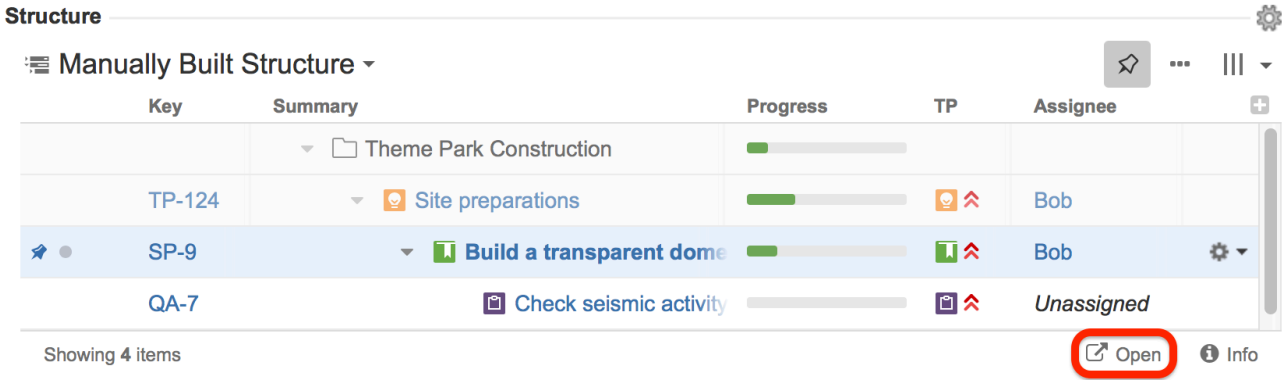
- Search and the Secondary panel is unavailable
- The toolbar is unavailable
- Only some fields can be edited from within Structure Gadget – see Editing from Gadget(see page 119)

However, many features are still available, including

- Most keyboard shortcuts are functional
- You can rearrange issues with drag-and-drop, if editing is enabled
- You can move, create, edit, and delete issues using the keyboard, if editing is enabled
- You can add, remove and rearrange columns
- If the gadget is displayed in its "home" Jira dashboard (not in Confluence or elsewhere), the last column lets you use the action drop-down for issues

3.10.1.5 Open on Structure Board

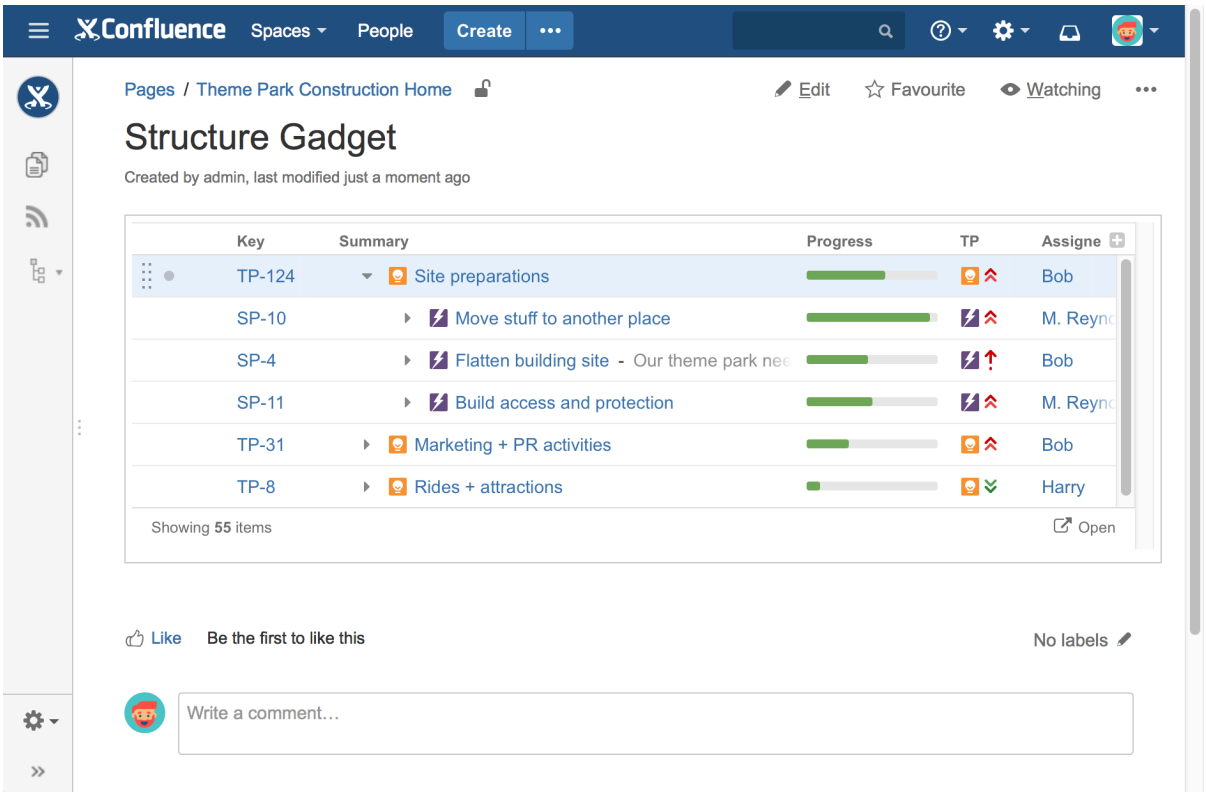
Working from the Structure Board(see page 83) provides the most unrestricted Structure experience. To get to the Structure Board from a gadget, click the **Open** link.



The structure will open on the Structure Board with the same filters[43] and transformations[44] that were applied in the original gadget. For example, if the gadget only shows items from a specific project, sorted by Assignee, that's exactly what you'll see on the Structure Board. To review or remove these transformations, click the Transformations button in the panel toolbar.

3.10.2 Confluence Gadget

You can embed a Structure gadget into a Confluence page to view the structure in Confluence.

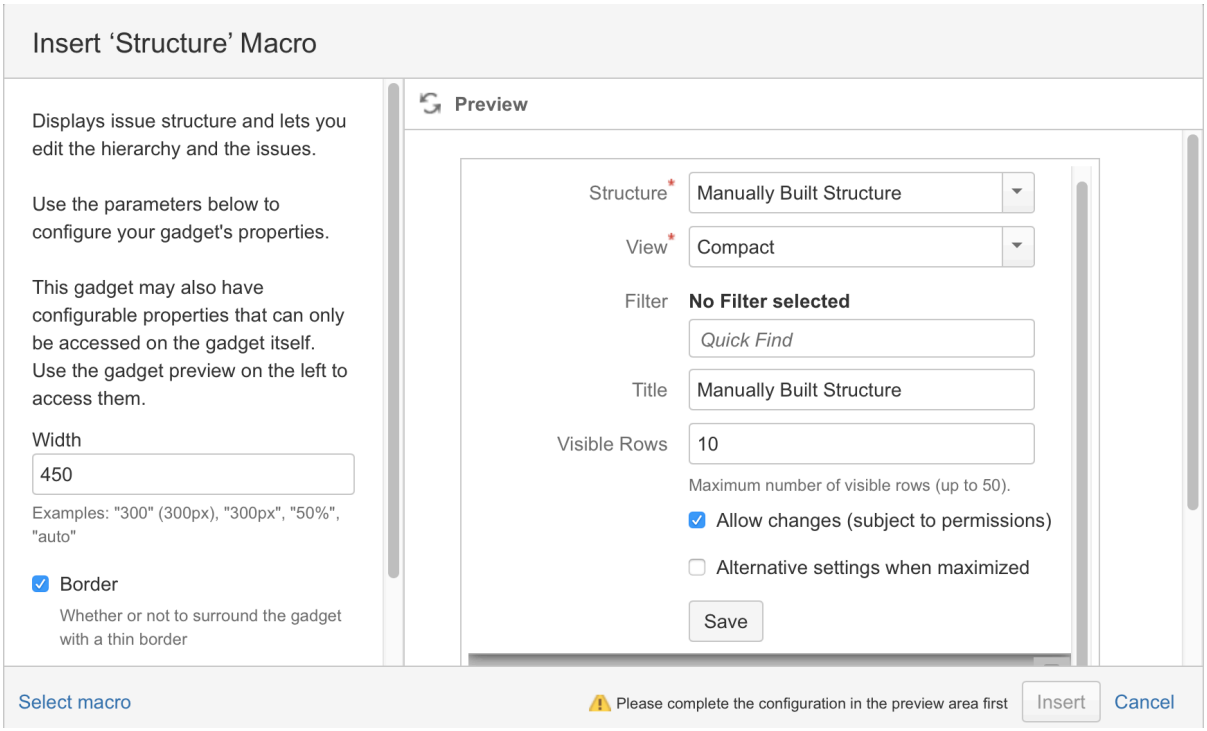


43 https://wiki.almworks.com/display/structure2gmaster/Filter
44 https://wiki.almworks.com/display/structure2gmaster/Transformations

⚠ Before you can use a Structure gadget on a Confluence page, your Confluence administrator must add Structure gadget to the Confluence configuration(see page 541). If you try to insert a macro and don't see *Structure* in the available list, most likely the gadget is not configured.

3.10.2.1 Adding a Structure Gadget to a Confluence Page

When editing a Confluence page, click Insert/Edit Macro, and select **Structure**. The macro configuration dialog appears.



Select your gadget configuration:

1. Select a **Structure**. Click arrow down in the Structure selector to view recently used and favorite structures, or start typing the structure name and let the drop-down suggest matching structures.
2. Select a **View**. Click arrow down to choose from views associated with the selected structure, start typing and let Structure suggest matching views, or create a new view. The selected view(see page 336) determines which columns the gadget displays. (You will be able to adjust the view later.)
3. Optionally, configure a **Filter**. The displayed structure will be filtered in the same way as in the Structure Board – see Filter(see page 183). You can choose between a text filter, JQL, S-JQL, or a saved JQL filter – see Search(see page 180).
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.

3.10.2.2 Using the Gadget

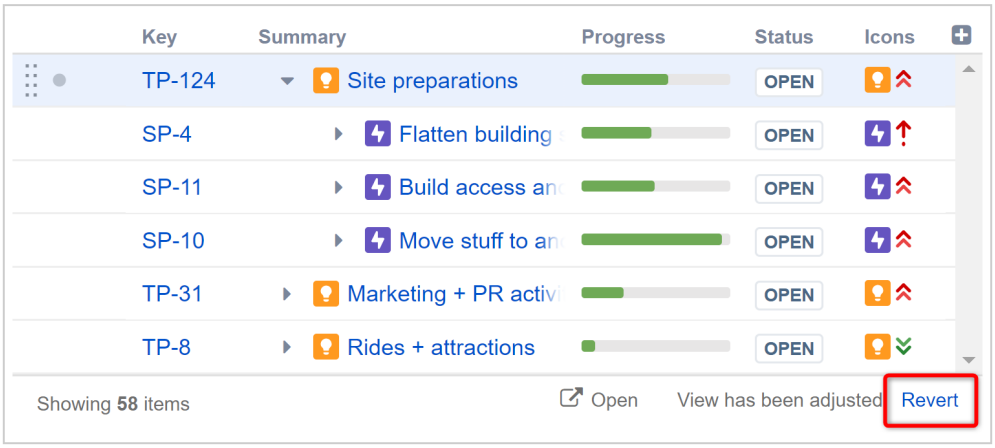
When viewing a structure in a gadget, some of Structure's features will be limited or unavailable:

- Search and the Secondary panel is unavailable
- The toolbar is unavailable
- The displayed Structure gadget is not suitable for printing. Support for a printable Structure gadget is coming later. For now, please use Printable Page(see page 380) to print a structure separately.
- In the current version the Structure Gadget in Confluence is supported, but impossible to edit.

3.10.2.3 Customizing Gadget View

Once you have created your gadget, you can customize by adding, removing or rearranging columns – see Customizing Columns(see page 331).

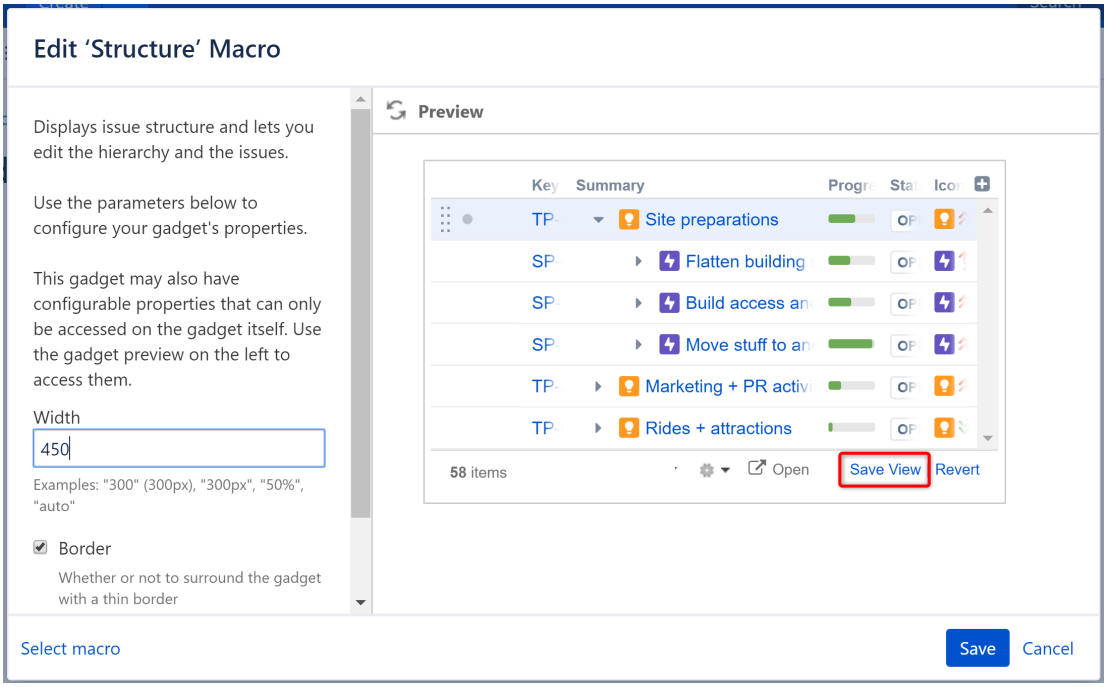
Once you've made changes to the gadget's view, a new indicator will appear at the bottom of the gadget, along with option revert back to the view's original setting.



In order to save the view, you need to enter Edit mode for the Confluence page and then edit the Structure macro. From there, you have a couple of options for saving your changes.

Save

Click the **Save** link to apply all the changes you have made to the existing view. The view will be updated everywhere it is in use, in the gadget, on Structure Board, etc. It will also impact all other users with access to that view.

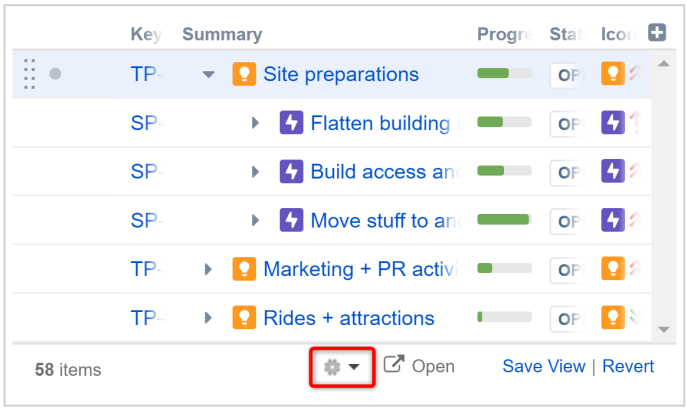


You must have Update permission(see page 346) for the current view to save changes.

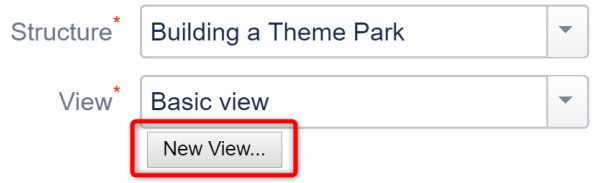
Save As

There is no Save As option on the Confluence gadget; however, it can be accomplished using the following method:

1. Adjust the gadget's view by adding, removing or rearranging columns – see Customizing Columns(see page 331) for details.
2. From the Edit 'Structure' Macro screen, open the gadget configuration by clicking the edit icon.



3. Click the **New View** button, located beside the view selector. An additional form appears – enter the new view name and click **Create View**.



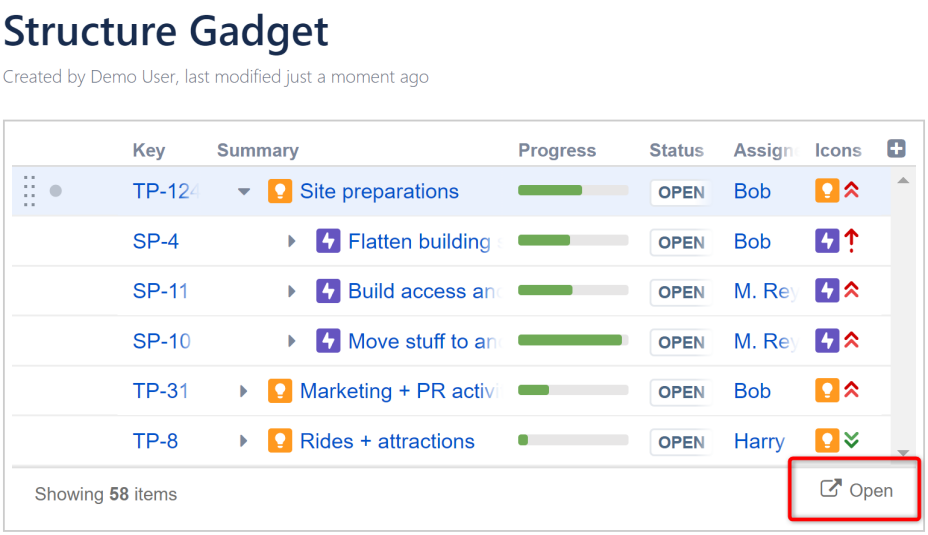
4. If this gadget is going to be visible to other users, make sure they have access to the view you've created by clicking **Let everyone use this view**.

Use this method if you don't have **Update** access to the original view or want to continue using the original view elsewhere.

If the user viewing the gadget does not have **Use** permission on the configured view, the gadget will show a default view with only Issue Key and Summary as columns.

3.10.2.4 Open on Structure Board

Working from the Structure Board(see page 83) provides the most unrestricted Structure experience. To get to the Structure Board from a gadget, click the **Open** link.



The structure will open on Structure Board with the same filters[45] and transformations[46] that were applied in the original gadget. For examples, if the gadget only shows items from a specific project, sorted by Assignee, that's exactly what you'll see on the Structure Board. To review or remove these transformations, click the Transformations button in the panel toolbar.

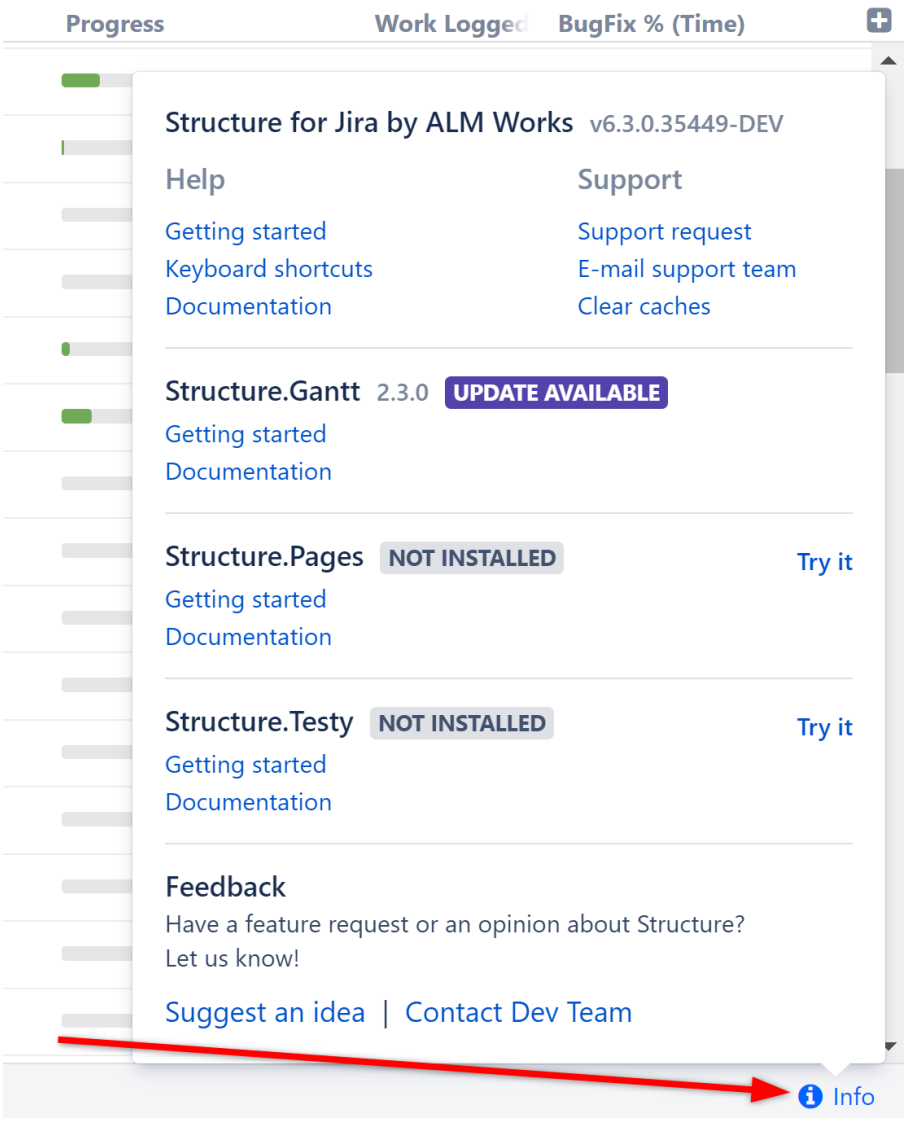
3.11 Getting Help

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[47] or contact our support team.

3.11.1 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 documentation and keyboard shortcuts, version and update information for Structure and Structure extensions, and links for our support services.

45 https://wiki.almworks.com/display/structure2gmaster/Filter
46 https://wiki.almworks.com/display/structure2gmaster/Transformations
47 https://wiki.almworks.com/display/structure2gmaster/Structure+User%27s+Guide



3.11.2 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[48]
- Suggest an idea[49] on our UserVoice forum
- Write to us[50] just to say hi or with any comments or questions

3.12 Integrations

- Better PDF Exporter for Jira(see page 405)
- ScriptRunner(see page 405)

48 http://almworks.com/structure/support-request
49 http://almworks.com/structure/suggest-idea
50 mailto:structure@almworks.com?subject=About%20Structure

- Tempo Timesheets(see page 420)
- Xporter(see page 421)
- Xray Test Management for Jira(see page 422)
- Power Scripts for Jira(see page 422)

3.12.1 Better PDF Exporter for Jira

Better PDF Exporter for Jira[51] allows you to make customizable PDF exports from Jira. When installed with Structure, it allows you to create highly-customized PDF exports from any structure.

3.12.1.1 Key benefits to using Better PDF Exporter with Structure

- Fully customize your PDF export using their predefined templates or by designing your own export process
- Export automated structures to PDF - all generators are run prior to the export
- Support for Structure-specific items: folders, memos, and generators
- Build PDFs with embedded attachments, bookmarks, charts and more
- Export structures to time sheets, invoices, Release Notes, IEEE-standard Requirements Specification docs and more
- Automate the export process using Automation for Jira, ScriptRunner or RestAPI

Learn more: Better PDF Export for Jira - Structure Integration[52]

3.12.2 ScriptRunner

ScriptRunner is an app by Adaptavist[53] 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[54].

3.12.2.1 Use-cases

- Creating (multiple) New Structures Programmatically(see page 410)
- Creating Generators with ScriptRunner(see page 411) - This script can be used to create new generators in a structure.
- Updating a field (ex. label) when checking all issues against a JQL query(see page 420) - This script will update a field (in this case Labels) for all issues that pass a certain JQL query.
- Automatically Remove Issues Based on JQL Query(see page 406) - This script will remove issues that do not satisfy a certain JQL query from a specific manually-built structure.
- Show All Structure Boards and Corresponding Item Counts(see page 415) - This script will show you the list of structures which exist in the instance and how many items were added to them manually. The result will be shown below the console.
- Bulk Change Owners of Structures and Generators(see page 408) - 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.

51 https://marketplace.atlassian.com/apps/5167/better-pdf-exporter-for-jira
52 https://www.midori-global.com/products/better-pdf-exporter-for-jira/server/documentation/integrations/structure
53 https://scriptrunner.adaptavist.com/latest/index.html
54 https://library.adaptavist.com

- Show work logged per user and issue for a structure(see page 418) - This script will create a table showing users, each issue they logged work on and how many hours were logged for each particular issue.

3.12.2.2 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 =
'16.10.0')

@WithPlugin("com.almworks.jira.structure")

@PluginModule
StructureComponents structureComponents

def plugin =
com.atlassian.jira.component.ComponentAccessor.pluginAccessor.getPlugin('com.alm
works.jira.structure')
```

```
def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def permission = PermissionLevel.valueOf("ADMIN")



// Here we are going to get our structure and then get the forest that it is
built on
def structureName = "name1"
if (structureManager.getStructuresByName(structureName, permission).isEmpty()){
    log.warn "Something went wrong, couldn't find the structure."
 return
}
def struct = structureManager.getStructuresByName(structureName, permission)[0]
def forestSpec = ForestSpec.structure(struct.getId())
def forestSrc = forestService.getForestSource(forestSpec)
RowManager rowManager = structureComponents.getRowManager()
def forest = forestSrc.getLatest().getForest()


// This will allow us access to useful helper functions, in this case the
application of our JQL query to our rows.
def helper = plugin.getModuleDescriptor('helper').module


// This variable will hold all our issues that match our JQL query.
def matchingIssues = new LongOpenHashSet()
// This variable will store all the elements in our structure that are issues
(vs folders or generators).
LongArray onlyIssues = new LongArray()
// This variable will hold the rows that that are eventually removed from the
Structure.
LongArray matchingRows = new LongArray();


// This will turn our JQL string into a Jira query.
def jqlQuery = "assignee = Eve"
Query query =
JiraComponents.getComponent(com.atlassian.jira.jql.parser.JqlQueryParser).parseQ
uery(jqlQuery);
```

```
// Here we are iterating over all the rows of our structure to get the issues.
for (LongIterator ri : forest.getRows()) {
    StructureRow row = rowManager.getRow(ri.value())
    ItemIdentity itemId = row.getItemId()
    if (CoreIdentities.isIssue(itemId)) {
        onlyIssues.add(itemId.getLongId())
    }
}

// Here we are evaluating which items match the query. If we change the boolean
value to false, we get the opposite.
helper.matchIssues(onlyIssues, query, true, matchingIssues);

// Now we are iterating over our structure again, row by row, to translate the
issues that we want to remove to their respective rows.
for (LongIterator ri : forest.getRows()) {
    StructureRow row = rowManager.getRow(ri.value())
    ItemIdentity itemId = row.getItemId()
    if (CoreIdentities.isIssue(itemId) &&
matchingIssues.contains(itemId.getLongId())) {
        matchingRows.add(ri.value())
    }
}

// Here we pass the rows we identified as unwanted to our remove function.
forestSrc.apply(new
ForestAction.Remove(matchingRows.subList(0,matchingRows.size())))
```

3.12.2.3 Bulk Change Owners of Structures and Generators

```
import com.atlassian.jira.ComponentManager
import com.atlassian.jira.user.ApplicationUsers
import org.apache.log4j.Category

def oldOwnerKey = 'admin'
def newOwnerKey = 'ragnar'
```

```
def resync = true // do full resync after update

def plugin = ComponentManager.getInstance().getPluginAccessor().getPlugin('com.a
lmworks.jira.structure')
def loader = plugin.getClassLoader()
def StructureAuth = loader.loadClass('com.almworks.jira.structure.api.auth.Struc
tureAuth')
def JiraUser = loader.loadClass('com.almworks.jira.structure.api.permissions.Per
missionSubject$JiraUser')
def structureManager = plugin.getModuleDescriptor('structure-
manager').getModule()
def syncManager = plugin.getModuleDescriptor('sync-manager').getModule()

def oldOwner = JiraUser.newInstance(oldOwnerKey)
def newOwner = JiraUser.newInstance(newOwnerKey)
def newOwnerUser = ApplicationUsers.byKey(newOwnerKey)
if (newOwnerUser == null) {
    def message = "Cannot find user by user key: $newOwnerKey"
 log.error(message)
    return message
}

// The actual work is done here

def changedStructures = []
def changedSynchronizers = []
def success = false
def exception = null

try {
    StructureAuth.sudo {
        structureManager.getAllStructures(null, true).each { st ->
            // Change owner
 if (st.owner == oldOwner) {
                st.owner = newOwner
                st.saveChanges()
                changedStructures << st
            }
            // Change owner of synchronizers installed for this structure
 syncManager.getInstalledSynchronizersForStructure(st.id).each { sync ->
                if (sync.userKey == oldOwnerKey) {
                    def enabled = syncManager.isAutosyncEnabled(sync.instanceId)
                    if (enabled) {
                        syncManager.setAutosyncEnabled(sync.instanceId, false)
                        syncManager.updateSynchronizer(sync.instanceId,
sync.getParameters(), newOwnerUser)
                        if (resync) {
                            syncManager.resync(sync.instanceId, true, null)
                        } else {
                            syncManager.setAutosyncEnabled(sync.instanceId, true)
                        }
                    } else {
                        syncManager.updateSynchronizer(sync.instanceId,
```

```
sync.getParameters(), newOwnerUser)
                    }
                    changedSynchronizers << sync
                }
            }
        }
    }
    success = true
} catch (Exception e) {
    log.warn("Failed to change owner from '$oldOwnerKey' to '$newOwnerKey'", e)
    exception = e
}

// Output message about changed structures and synchronizers to the log and to
the console output
def msg = "Script to change owner from '$oldOwnerKey' for '$newOwnerKey' " +
        (success ? "finished successfully" : "failed (${exception &&
exception.message})") + "\n" +
        "Changed structures:\n" + changedStructures.collect({ "#${it.id} $
{it.name}" }).join("\n") + "\n" +
        "Changed synchronizers:\n" + changedSynchronizers.collect({ "#$
{it.instanceId} (for structure #${it.structureId})" }).join("\n")

log.warn(msg)
msg.replaceAll("\n", "<br>")
```

3.12.2.4 Creating a New Structure Programmatically

Running the following script you can create a new structure.

```
package examples.docs.structure

import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version =
'16.9.0')

@WithPlugin("com.almworks.jira.structure")

@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()

def permission = PermissionLevel.valueOf("ADMIN")

// It is important to check if an existing structure already exists; otherwise, this will
```

```
create a new structure with the same name.
if (structureManager.getStructuresByName("testScript", permission).isEmpty()){
log.warn "no existing Structure found"
// It is important to note that without the saveChanges() call at the end, the new
structure won't be saved to the database and will disappear after the script is run.
structureManager.createStructure().setName("testScript").saveChanges()
} else {
log.warn "found a pre-existing structure"
}
```

3.12.2.5 Creating Generators

Below are a series of scripts that run through the creation of an Insert, Extend, Sort and Filter Generator.

JQL Inserter

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.generator.CoreGeneratorParameters
import java.util.Map

@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureName = "test" // name of the structure you want to add this
generator to
def jql = 'project = stmb and type = story' // this is the JQL query you want
the generator to executre

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def generatorManager = structureComponents.getGeneratorManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureId = structureManager.getStructuresByName(structureName,
permission)[0].getId()

// inserter
def params = new java.util.HashMap()
params.put(CoreGeneratorParameters.JQL, jql)
def jqlinserterId = generatorManager.createGenerator("com.almworks.jira.structur
e:inserter-jql", params, structureId)
```

```
def generatorItem = CoreIdentities.generator(jqlinserterId); // item to add to
forest
def forestSource =
forestService.getForestSource(ForestSpec.structure(structureId)) //resolving
forest source for structure
forestSource.apply(new ForestAction.Add(generatorItem, 0, 0, 0))
```

Extend stories under epics and extend by blocking link

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.item.ItemForestBuilderImpl;
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.generator.CoreStructureGenerators
import com.atlassian.jira.issue.link.IssueLinkTypeManager
import com.almworks.jira.structure.api.generator.CoreStructureGenerators
import java.util.Map

@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureName = "test"
def structureId = structureManager.getStructuresByName(structureName,
permission)[0].getId()
def forestBuilder = new ItemForestBuilderImpl()
def generatorManager = structureComponents.getGeneratorManager()

def epicsExtenderItem =
generatorManager.createGenerator(CoreStructureGenerators.EXTENDER_AGILE, [:],
null)
forestBuilder.nextRow(CoreIdentities.generator(epicsExtenderItem))

def issueLinkTypeManager = ComponentAccessor.getComponent(IssueLinkTypeManager)
def blocksLinkTypes = issueLinkTypeManager.getIssueLinkTypesByName('Blocks')
def blocksLinkTypeId = blocksLinkTypes[0].id

def params = [
        'linkTypeId': blocksLinkTypeId, // id of the Blocks issue link type
 'direction': 'outward',         // direction of the link type, could be either
```

```
'inward' or 'outward'
 'disableActions': false,        // enable structure actions, i.e. issues DnD
 'from': 2,
        'to': 2 // number of levels to extend issue at
]
def blocksExtenderItem =
generatorManager.createGenerator(CoreStructureGenerators.EXTENDER_LINKS, params,
structureId)
forestBuilder.nextRow(CoreIdentities.generator(blocksExtenderItem))

def forestService = structureComponents.getForestService()
def forestSource =
forestService.getForestSource(ForestSpec.structure(structureId))
def forestToAdd = forestBuilder.build()
forestSource.apply(new ForestAction.Add(forestToAdd, 0, 0, 0))
```

Sorting by a formula result and manual sorter

```
package examples.docs.structure

import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin
import com.atlassian.jira.component.ComponentAccessor
import com.almworks.jira.structure.api.structure.Structure
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.item.ItemForestBuilderImpl;
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.generator.CoreStructureGenerators

@WithPlugin("com.almworks.jira.structure")
@PluginModule
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def permission = PermissionLevel.valueOf("ADMIN")
def structureName = "test"
def structureId = structureManager.getStructuresByName(structureName,
permission)[0].getId()
def forestBuilder = new ItemForestBuilderImpl()
def generatorManager = structureComponents.getGeneratorManager()

def manualSorterItem =
generatorManager.createGenerator(CoreStructureGenerators.SORTER_MANUAL, [:],
null)
forestBuilder.nextRow(CoreIdentities.generator(manualSorterItem))
/* adding formula-based sorter
*/
def formulaSorterParams = [
```

```
        attribute: [
                id : "expr",
                format: "order",
                params : [
                        formula: "assignee + key",
                        variables: [
                                assignee: [ id: "Assignee", format: "text"],
                                key: [ id: "Key", format: "text"]
                        ]
                ]
        ]
]
def formulaSorterItem =
generatorManager.createGenerator(CoreStructureGenerators.SORTER_ATTRIBUTE,
formulaSorterParams as Map, null)
forestBuilder.nextRow(CoreIdentities.generator(formulaSorterItem))
def forestService = structureComponents.getForestService()
def forestSource =
forestService.getForestSource(ForestSpec.structure(structureId))
def forestToAdd = forestBuilder.build()
forestSource.apply(new ForestAction.Add(forestToAdd, 0, 0, 0))
```

Below adds a JQL filter to a structure

```
package examples.docs.structure

// Structure imports
import com.almworks.jira.structure.api.forest.ForestSpec
import com.almworks.jira.structure.api.forest.action.ForestAction
import com.almworks.jira.structure.api.generator.CoreGeneratorParameters
import com.almworks.jira.structure.api.item.CoreIdentities
import com.almworks.jira.structure.api.permissions.PermissionLevel
import com.almworks.jira.structure.api.StructureComponents

// Scriptrunner imports
import com.onresolve.scriptrunner.runner.customisers.PluginModule
import com.onresolve.scriptrunner.runner.customisers.WithPlugin

// Atlassian import (might be available without the import in some instances,
but better safe than sorry)
import com.atlassian.jira.component.ComponentAccessor

// A Hashmap
import java.util.Map

@Grab(group = 'com.almworks.jira.structure', module = 'structure-api', version =
'16.9.0')

@WithPlugin("com.almworks.jira.structure")

@PluginModule
```

```
StructureComponents structureComponents

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def generatorManager = structureComponents.getGeneratorManager()

// For brevity's sake we will have a permission variable that we pass when we
make an admin level request
def permission = PermissionLevel.valueOf("ADMIN")

// the false flag is optional, it is a variable whether we want to search
archived structures too
// we are getting the first (zeroth) element because this method returns a list,
be mindful if you have multiple structures with the same name!
def currentStructure = structureManager.getStructuresByName("testScript",
permission, false)[0]


def jqltext = "assignee=admin";
Map<String, Object> jqlparams = new java.util.HashMap()
jqlparams.put(CoreGeneratorParameters.JQL, jqltext)

// We have now added our JQL text to the filter (creating a different kind of
inserter works almost exactly the same way, just build up the params variable
differently)
def jqlfilterId = generatorManager.createGenerator("com.almworks.jira.structure:
filter-jql", jqlparams, currentStructure.getId())

def generatorItem = CoreIdentities.generator(jqlfilterId); // item to add to
forest
def forestSource =
forestService.getForestSource(ForestSpec.structure(currentStructure.getId())) //
resolving forest source for structure
forestSource.apply(new ForestAction.Add(generatorItem, 0, 0, 0))
```

3.12.2.6 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 =
'16.9.0')
@WithPlugin('com.almworks.jira.structure')

StructureComponents structureComponents =
```

```
ComponentAccessor.getOSGiComponentInstanceOfType(StructureComponents)
def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()

def structures = structureManager.getAllStructures(PermissionLevel.NONE)
def map = structures.collectEntries {
  def fs = forestService.getForestSource(ForestSpec.skeleton(it.id))
  [(it.name + ' (#' + it.id + ')'): fs.getLatest().getForest().size()]
}

map = map.sort {-it.value}

'<table><tr><th>Structure Name (ID)</th><th>Number of manually added rows</th></
tr>' +
map.collect {'<tr><td>' + Util.htmlEncode(it.key as String) + '</td><td>' +
it.value + '</td></tr>'}.join('')  +
  '</table>'
```

This script can be adopted to show the count of dynamically generated items by replacing ***ForestSpec.skeleton()*** with ***ForestSpec.structure()***. However, running this script will trigger all the structures and their generators, which can cause a performance impact. Be careful when running this in a production environment.

3.12.2.7 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 Management / Service Desk instance and a Dev instance, for example). The second script does the exact same thing for all linked issues on the same instance.

In order to run these scripts, there are a few preconditions that need to be met. You will need to create a scripted field[55] 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
```

55 https://scriptrunner.adaptavist.com/latest/jira/scripted-fields.html

```
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 +
otherIssue.getKey() + '</a>'
    count++
}
remIs
?:null
```

⚠ You should replace the value in `baseUrl` with the actual URL of your local instance.

3.12.2.8 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.getPlugin(
'com.almworks.jira.structure')

def structureManager = structureComponents.getStructureManager()
def forestService = structureComponents.getForestService()
def permission = PermissionLevel.valueOf("ADMIN")
def helper = plugin.getModuleDescriptor('helper').module

def struct = structureManager.getStructuresByName("test", permission)[0]
def structureName = struct.getName()
def forestSpec = ForestSpec.structure(struct.getId())
def forestSrc = forestService.getForestSource(forestSpec)
RowManager rowManager = structureComponents.getRowManager()
def forest = forestSrc.getLatest().getForest()

// this variable will store all the elements in our structure that are issues
(vs folders or generators).
```

```
LongArray onlyIssues = new LongArray()

// here we are iterating over all the rows of our structure to get the issues.
for (LongIterator ri : forest.getRows()) {
    StructureRow row = rowManager.getRow(ri.value())
    ItemIdentity itemId = row.getItemId()
    if (CoreIdentities.isIssue(itemId)){
        onlyIssues.add(itemId.getLongId())
    }
}

def wm = ComponentAccessor.getWorklogManager()
def issueManager = ComponentAccessor.getComponent(IssueManager)
def user = ComponentAccessor.jiraAuthenticationContext.getLoggedInUser()
def userToIssue = [:]
def seen = []

onlyIssues.each{ is ->
    def iss = issueManager.getIssueObject(is.value())
    def wl = wm.getByIssue(iss)

    def it = wl.each{
        def issue = iss.getKey()
        def author = it.getAuthorKey()
        def time = it.getTimeSpent()
        if (!seen.contains(author)){
            // put author in seen
 seen.push(author)
            def tempHM = [:]
            tempHM[issue] = time
            userToIssue[author] = tempHM
        } else {
            if(!userToIssue[author].containsKey(issue)){
                // create a new issue to time hashmap to add
 def tempHM = [:]
                tempHM[issue] = time
                userToIssue[author] += tempHM
            }
            else{
                userToIssue[author][issue] = userToIssue[author][issue] + time
            }
        }
    }
}

'<table><tr><th>Structure</th><th>User </th><th>Issue Worklogged (hours)</th></
tr>' +
        userToIssue.collect {'<tr><td>' + it.key + '</td><td><table>' +
                it.value.collect {'<tr><td>' + it.key + '</td> <td>' +
(it.value*0.000277778).toBigInteger() + '</td></tr>' }.join('') +
                '</table></td></tr>'}.join('') +
        '</table>'
```

3.12.2.9 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(),
PagerFilter.getUnlimitedFilter())
        searchResult.issues.each{issue -> labelManager.addLabel(user, issue.id,
labelName, false)
        }
    }
}
```

3.12.3 Tempo Timesheets

If you're using the Tempo Timesheets app[56], our integration allows you to track all your KPIs per account within a structure.

56 https://marketplace.atlassian.com/apps/6572/tempo-timesheets-time-tracking-report

3.12.3.1 Grouping by Tempo Accounts

1. Add tasks, either manually or using Structure Automation(see page 122).
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.

3.12.4 Xporter

Xporter[57] allows you easily export Jira issues to PDF, Excel and Word.

3.12.4.1 Key benefits to using Xporter with Structure

- Export structures directly from the Structure board
- Export structures from the issue screen
- Export children on any row
- Extract data from specific rows
- Export the children count for issues in each row

Learn more: Xporter - Structure Integration[58]

57 https://marketplace.atlassian.com/apps/891368/xporter-export-issues-from-jira
58 https://confluence.xpand-it.com/display/public/XPORTER/Structure+for+Jira+-+Projects+at+Scale

3.12.5 Xray Test Management for Jira

Xray[59] provides automated test management for Jira.

3.12.5.1 Key benefits to using Xray with Structure

- Include Xray Test and Test Sets within the Structure hierarchy
- Add/remove Tests and Test Sets by dragging them within the structure
- Visualize and manage associations between Tests and Requirements within Structure

Learn more: Xray - Structure Integration[60]

3.12.6 Power Scripts for Jira

Power Scripts™ for Jira[61] provides custom Jira scripting and automation capabilities to automate workflows and complex tasks.

3.12.6.1 Key benefits to using Power Scripts with Structure

- Add scripted content as a column within the structure
- Use scripts to display specific issue dependencies within the structure

Learn More: Power Scripts™ + Structure for Jira Combined Simplify Dependency Management[62]

59 https://marketplace.atlassian.com/apps/1211769/xray-test-management-for-jira?hosting=server&tab=overview
60 https://docs.getxray.app/display/XRAY/Integration+with+Structure
61 https://marketplace.atlassian.com/apps/43318/power-scripts-jira-script-automation?hosting=server&tab=overview
62 https://www.cprimeapps.com/power-scripts-structure-for-jira-combined-simplify-dependency-management/

4 How-To Articles

The following articles walk you through specific daily uses cases, and how Structure can be used to solve them.

- Calculate the Cost to Complete Issues and Projects(see page 423)
- Update Assignees to Match Parent Issues(see page 428)
- Calculate Story Points for Epics and Write the Values to Jira(see page 433)
- Backlog Grooming for Teams(see page 438)
- Correcting Inconsistent or Incorrect Link Types with the Extend Generator(see page 440)
- Resource Allocation(see page 442)
- Enable SAFe (Scaled Agile Framework) with Structure(see page 445)
- Creating an Advanced Formula Column(see page 446)
- Track Overdue Tasks with Structure(see page 450)

4.1 Calculate the Cost to Complete Issues and Projects

4.1.1 Situation

You want to determine how much it will cost to complete each Story, Epic, or other grouping of issues, based on the amount of time remaining and the hourly rate of the assignee.

As a bonus, you would also like to make this information visible to users outside of Structure.

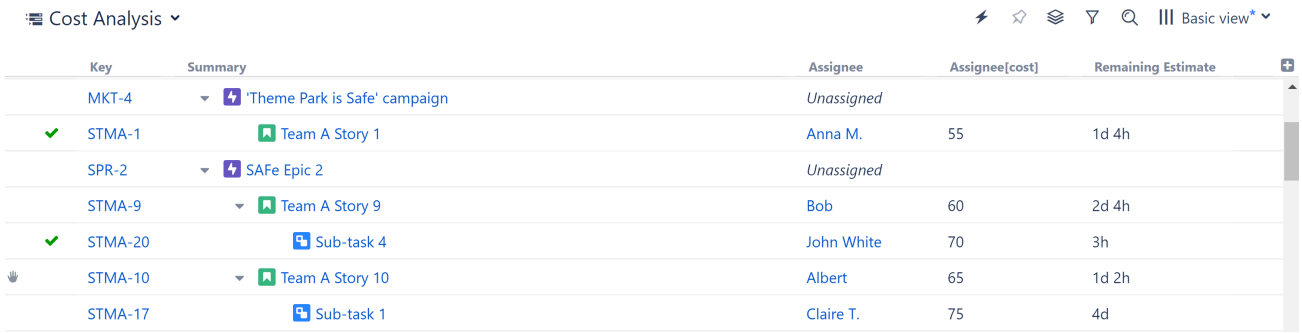
4.1.2 Solution

The following guide assumes you have:

- All stories and tasks assigned to specific Jira users
- A custom User property[63] containing each user's hourly rate of pay (we've named ours "cost")

4.1.2.1 Step 1: Add a Column for the Custom User Property

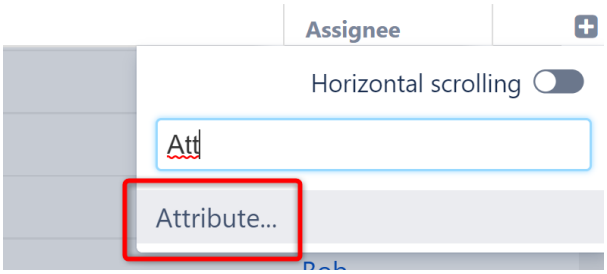
We've used Insert(see page 125) and Extend(see page 146) generators to build an Epic > Story hierarchy. We've also added a column showing the values from our custom user property, labeled **Assignee(cost)**.



Key	Summary	Assignee	Assignee[cost]	Remaining Estimate
MKT-4	'Theme Park is Safe' campaign	*Unassigned*		
STMA-1	Team A Story 1	Anna M.	55	1d 4h
SPR-2	SAFe Epic 2	*Unassigned*		
STMA-9	Team A Story 9	Bob	60	2d 4h
STMA-20	Sub-task 4	John White	70	3h
STMA-10	Team A Story 10	Albert	65	1d 2h
STMA-17	Sub-task 1	Claire T.	75	4d

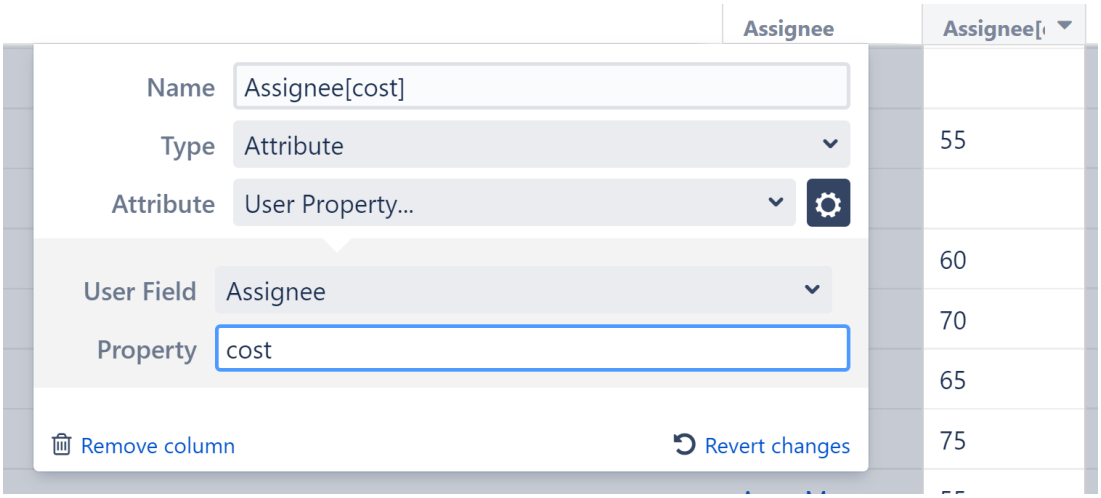
In order to add the Assignee(cost) column above, we had to add an Attribute column:

63 https://confluence.atlassian.com/adminjiraserver/create-edit-or-remove-a-user-938847025.html



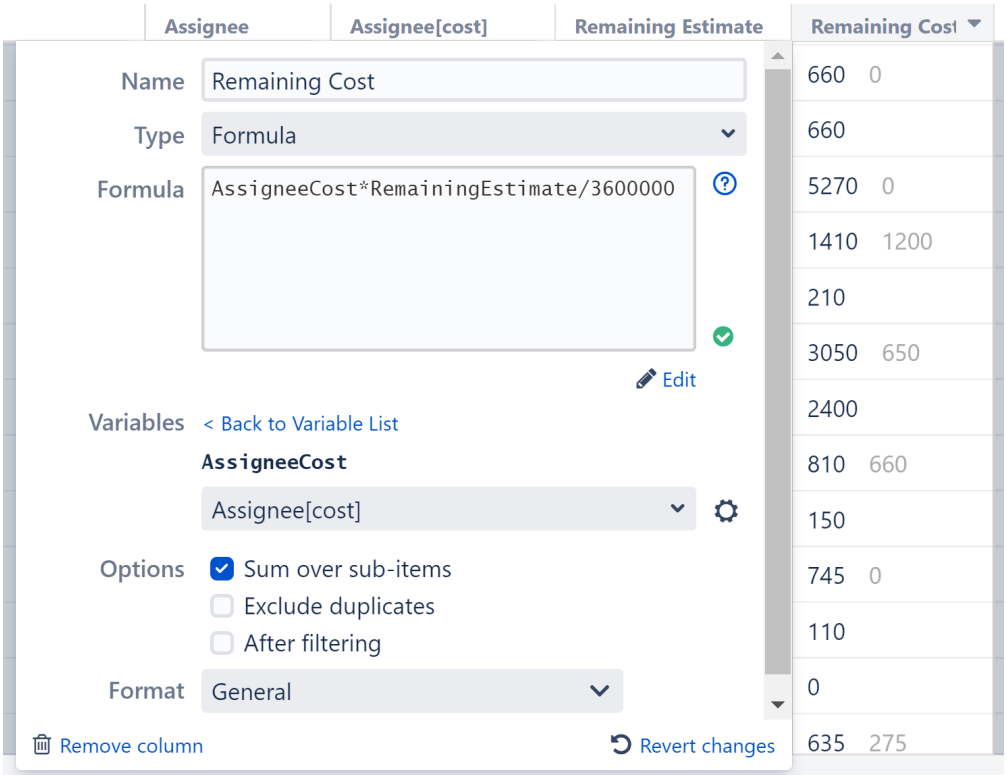
Under the column properties, enter the following values:

- **Attribute:** User Property...
- **User Field:** Assignee
- **Property:** Enter the key for the user property where you store each user's hourly pay rate - ours is called "cost"



4.1.2.2 Step 2: Calculate the Cost to Finish Each Issue

Next we'll add a formula column(see page 197) to calculate the cost to finish each issue in our structure.



In our formula, we're multiplying our assignee's hourly cost value by the remaining estimate (the estimate is in milliseconds, so we divided by 3,600,000 to find out how many hours of work are left).

- In the Variables list, make sure to assign your Assignee cost to the custom user property we discussed in Step 1. See Columns as Variables(see page 203) for more details.
- We want to roll our costs up to each epic, so we checked the Sum over sub-items box. If you don't need to do this, leave the box unchecked.
- If issues or tasks appear multiple times within an epic, you may want to exclude duplicates, to avoid an inflated total cost.

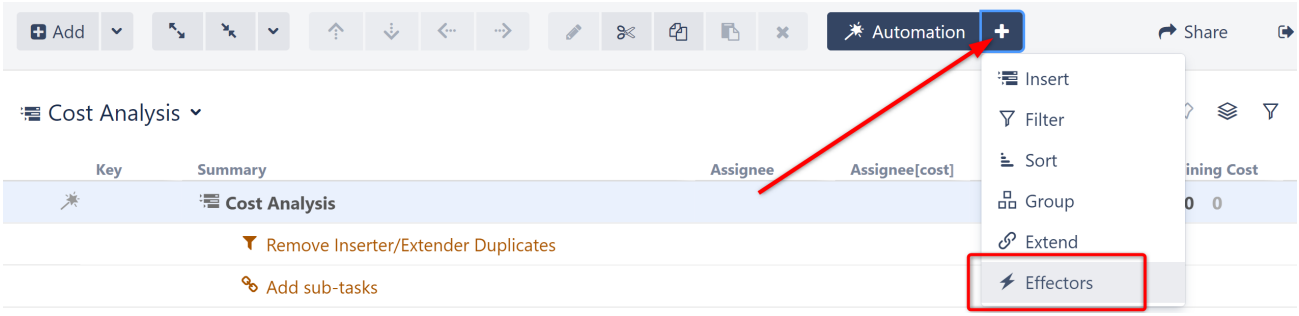
4.1.3 Bonus - Make This Information Visible Outside of Structure

Using Effectors(see page 154), we can write the values from our formula to a custom Jira field, making them visible to users outside of Structure.

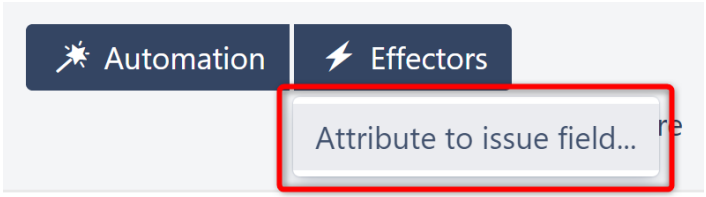
Before we start, you'll need to make sure you have a custom field where the Cost-to-Complete values can be stored. We created a custom field called "FinishCost".

4.1.3.1 Add an Effector

Open the Automation menu and select **Effectors**.

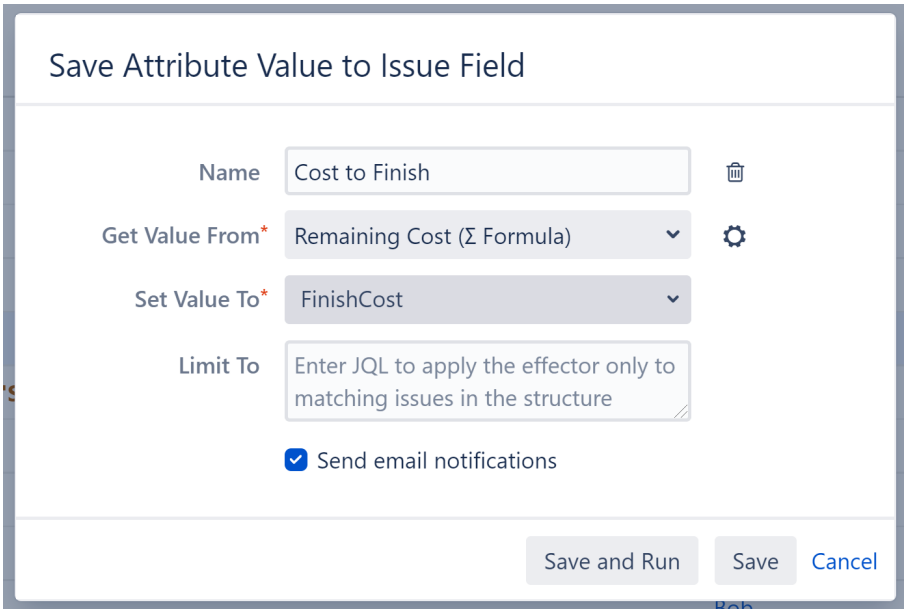


Choose **Attribute to issue field...**



On the Effector settings screen:

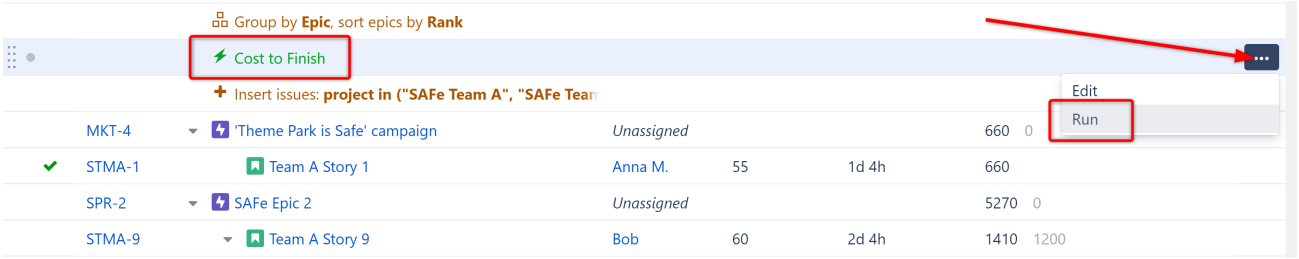
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Get Value From: select the formula column we added in Step 2
- Set Value To: select the custom Jira field you want to write the cost-to-complete values to
- Limit To: If you only want to write cost data for certain issues or issue types (for example, just for epics), enter the appropriate JQL. We've left this section blank, because we want to write values for all issues.
- Select whether email notifications should be sent when the Effector writes values to Jira



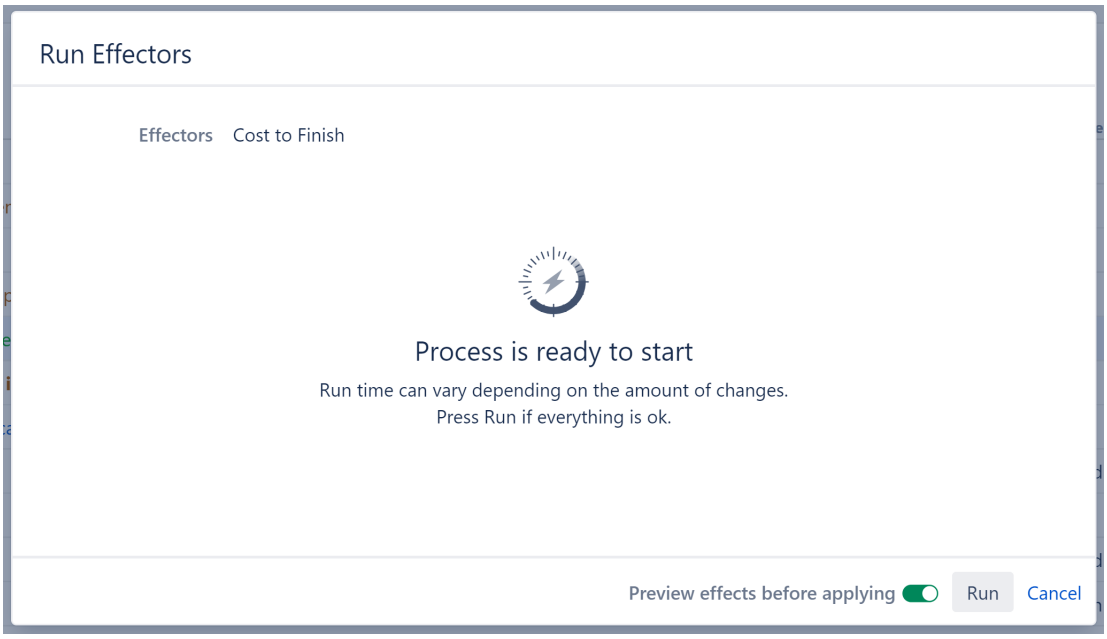
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

4.1.3.2 Run the Effector

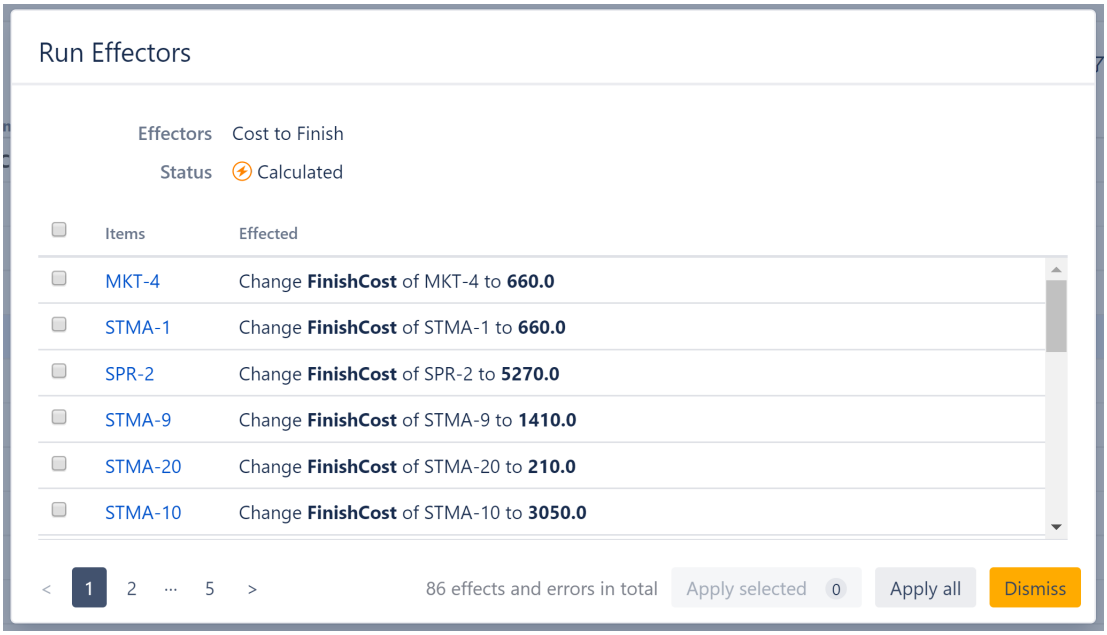
You can run an Effector directly from its settings screen (see above), or you can locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.



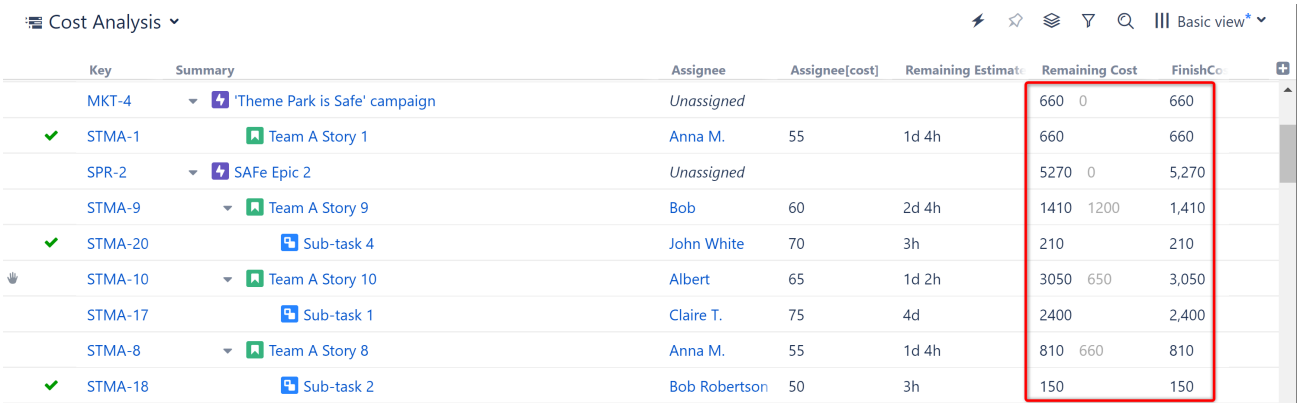
You have the option to **Preview effects before applying**. This allows you to view and approve every change that will be made by the Effector. Effectors update live Jira data, so we highly recommend using the preview option. Click **Run** to begin.



Once the preview is finished, you will see a list of changes that will be made by the Effector. You can select which changes you want made, or click **Apply All** to apply all changes.



Once you apply the selected changes, the cost values from your formula column will be written to the custom Jira field, where they can be viewed outside of Structure.



4.2 Update Assignees to Match Parent Issues

4.2.1 Situation

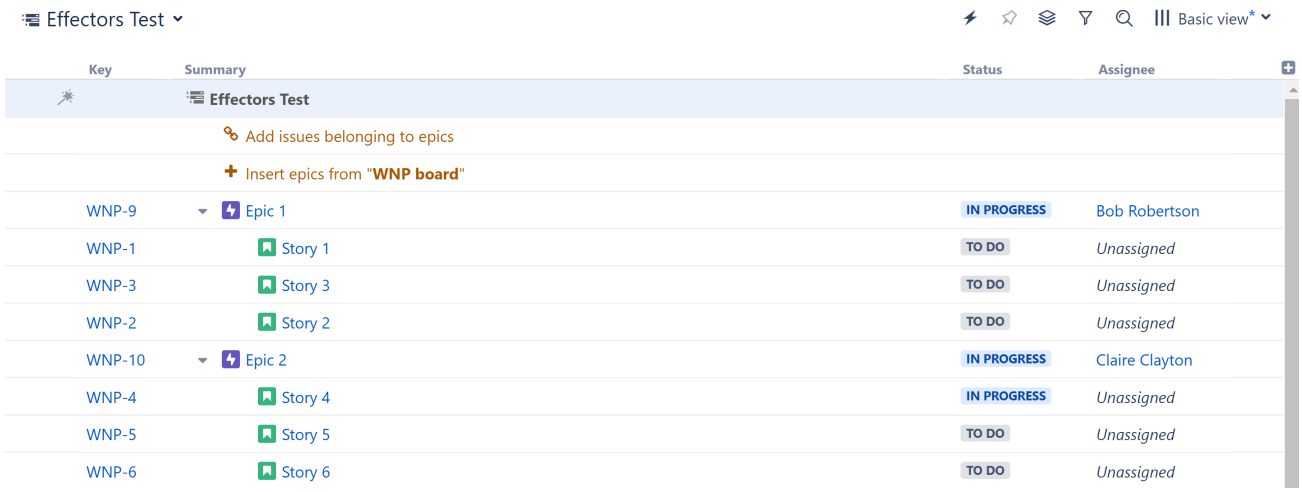
After arranging issues into a hierarchy, you want to automatically update the assignee for each sub-issue to match it's parent-issue. For example, make all stories within an epic share the same assignee as the epic.

4.2.2 Solution

We will use a simple formula(see page 197) and Effectors(see page 154) to update the Assignee field for each issue to match it's parent issue.

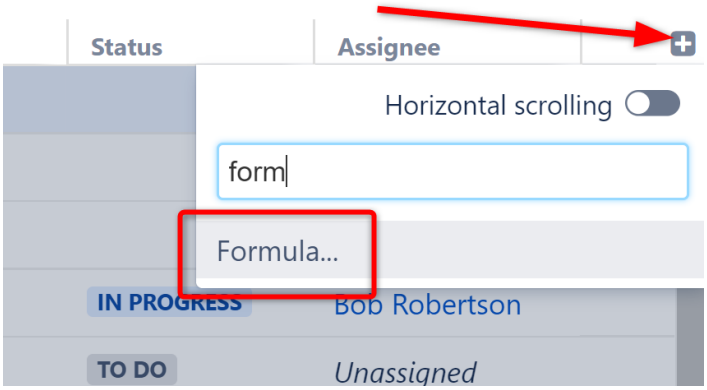
In this example, we've used Insert(see page 125)and Extend (see page 146)generators to build an Epic > Story hierarchy. Each epic has been assigned to a team member, but the stories are still unassigned. We're going to use the Attribute

to Issue Field Effector(see page 154), along with a Structure formula, to change the assignee of each story to match its corresponding epic.

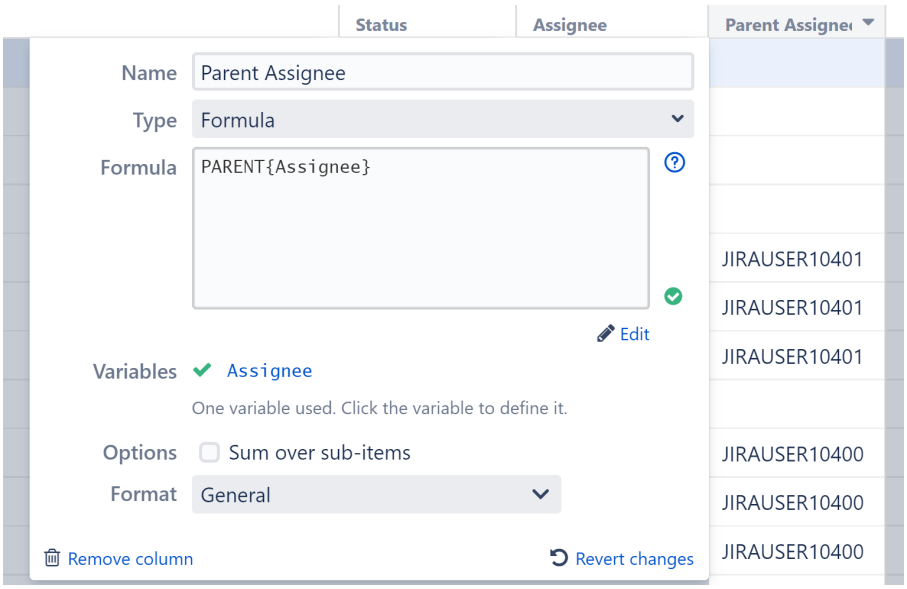


4.2.2.1 Step 1: Add a Formula Column to Capture the Parent's Assignee

Click the **+** icon to the right of the column headers, and select **Formula...**



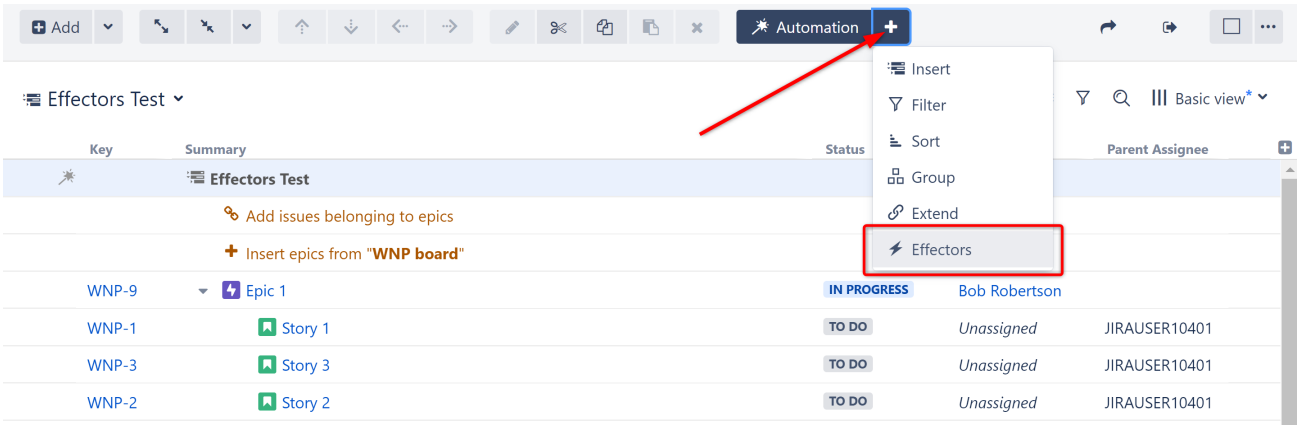
Give the column an appropriate name, and enter the following formula: `PARENT{Assignee}`



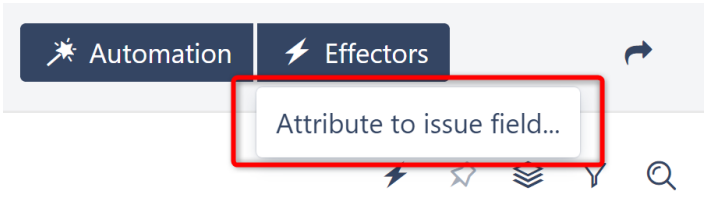
4.2.2.2 Step 2: Add an Effector

Now we need to take the information from our formula (which is only available in Structure) and write it to the Assignee field (so it is available anywhere in Jira).

Open the Automation menu and select **Effectors**.



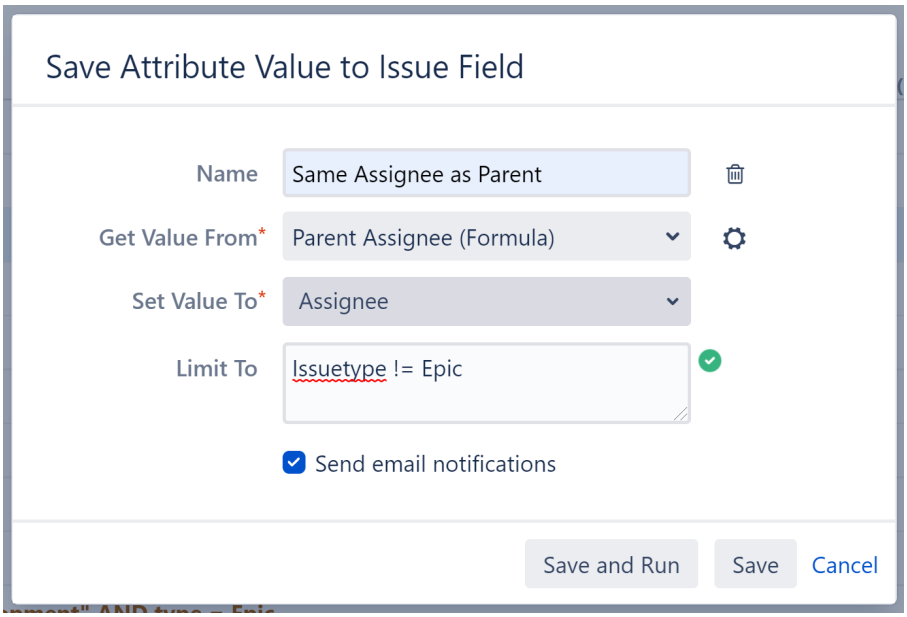
Choose **Attribute to issue field...**



On the Effector settings screen:

- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.

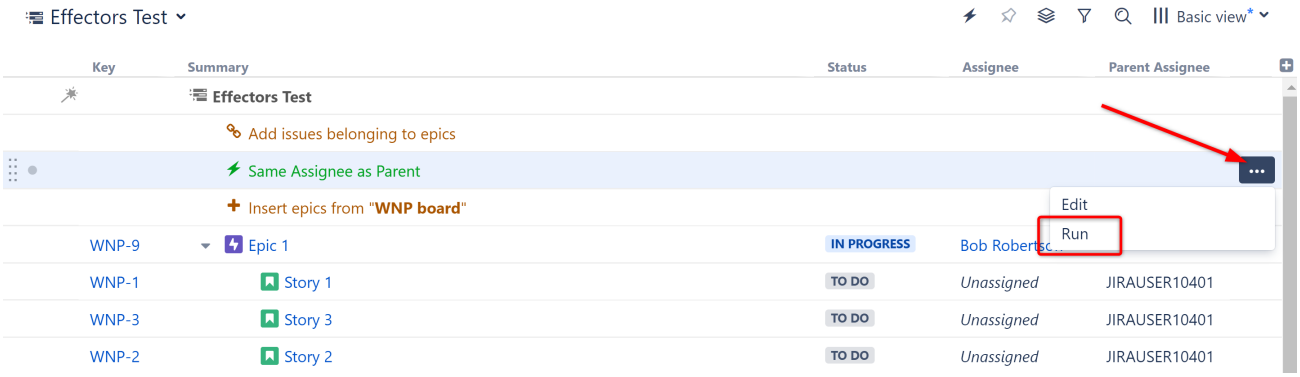
- Get Value From: Select the formula column we created in Step 1
- Set Value To: Assignee
- Limit To: Since we're getting our Assignee value from Epics, we want to make sure their Assignee value isn't changed - so we've added the following JQL Filter: `Issuetype != Epic`
- Select whether email notifications should be sent when the Effector writes values to Jira



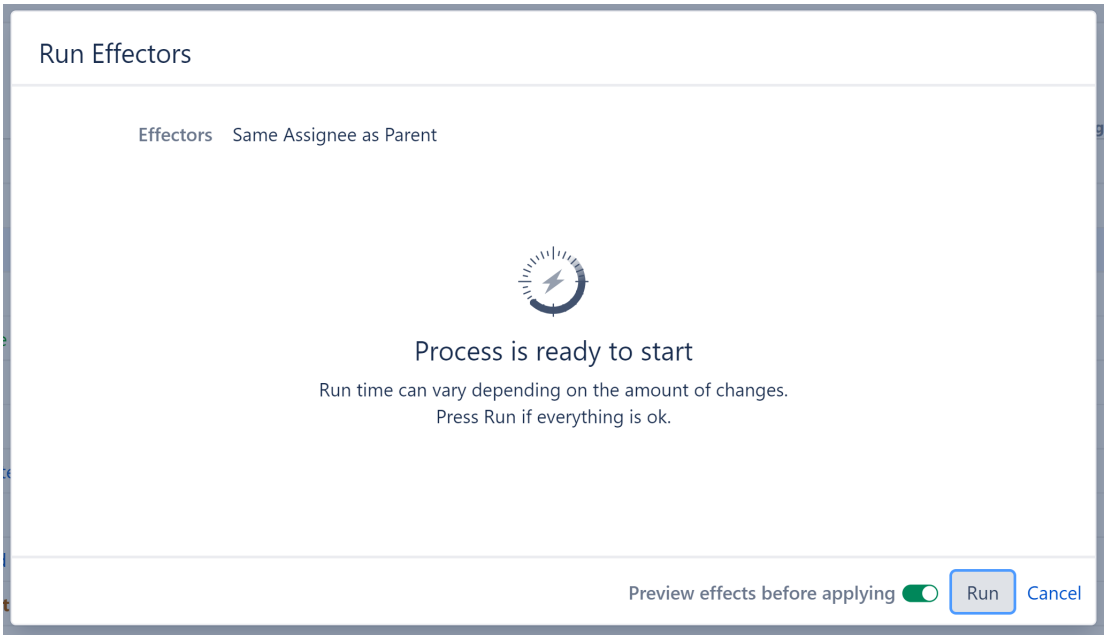
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

4.2.2.3 Step 3: Run the Effector

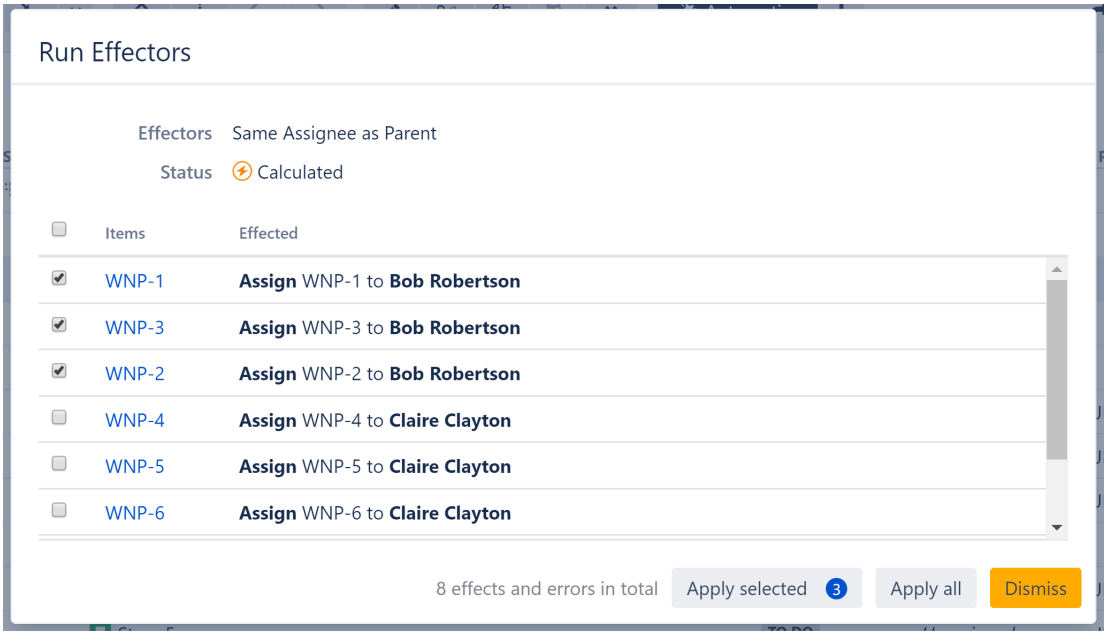
You can run an Effector directly from its settings screen (see above), or you can locate the Effector at the top of the structure, click the Action button (the three dots to the right of its row) and select **Run**.



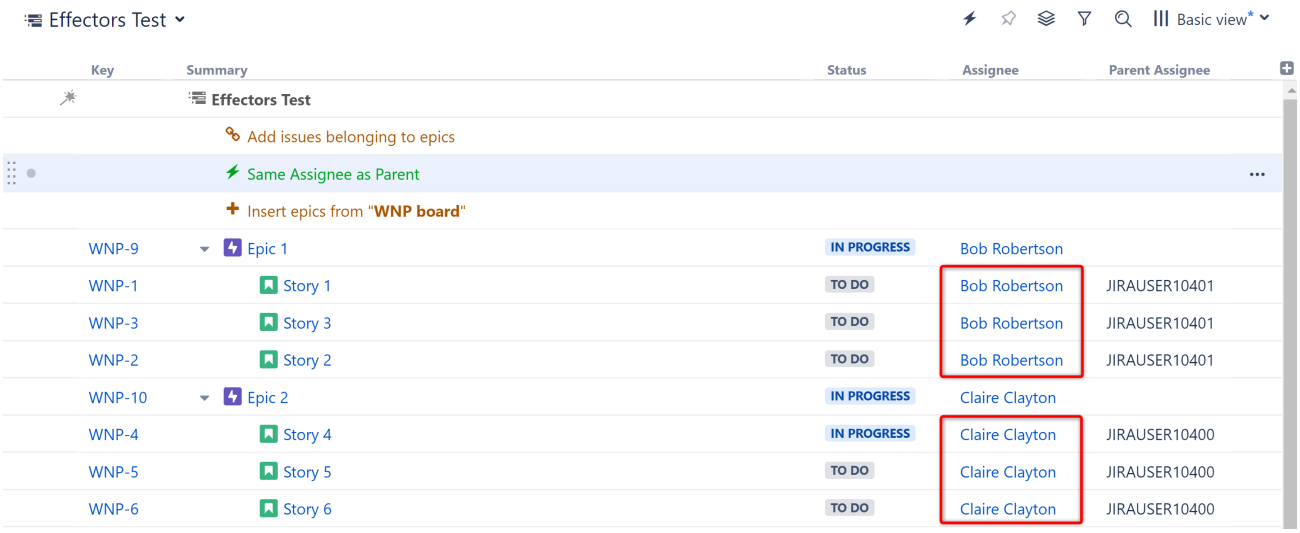
You have the option to **Preview effects before applying**. This allows you to view and approve every change that will be made by the Effector. Effectors update live Jira data, so we highly recommend using the preview option. Click **Run** to begin.



Once the preview is finished, you will see a list of changes that will be made by the Effector. You can select which changes you want made, or click **Apply All** to apply all changes.



Once you've applied the selected changes, the Assignee field for each child issue should be updated to match its parent's Assignee:



4.3 Calculate Story Points for Epics and Write the Values to Jira

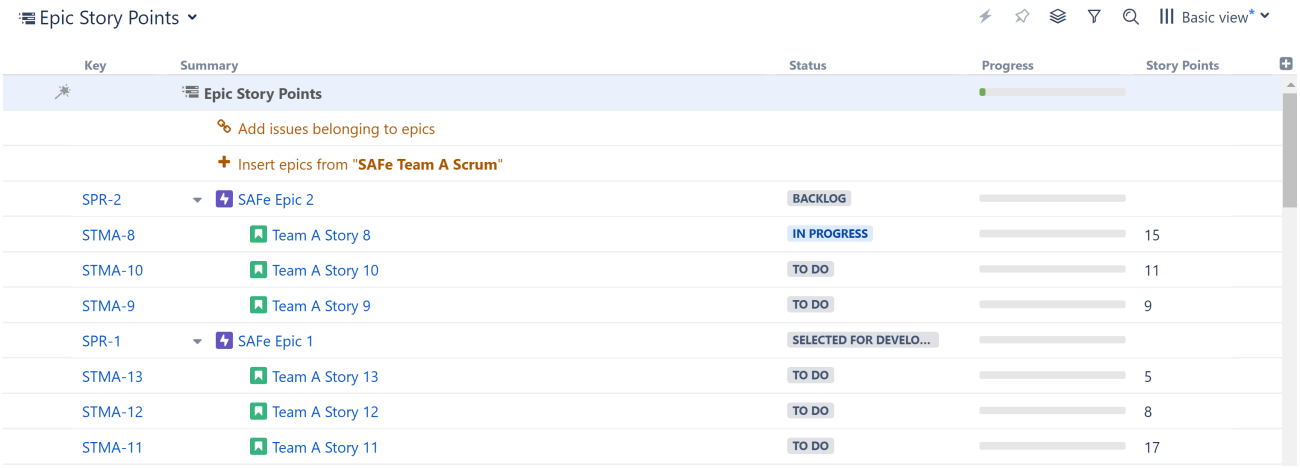
4.3.1 Situation

You want to update the Story Point value for each epic, based on the value of all the stories within the epic.

4.3.2 Solution

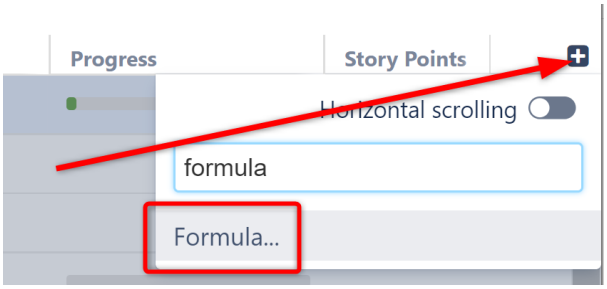
We will use a Formula column(see page 197) to calculate the total story points for each epic, and then use an Effector (see page 154)to update the epic's Story Point field.

In this example, we've used Insert(see page 125) and Extend(see page 146) generators to build an Epic > Story hierarchy. Notice that our Epics do not have Story Point values, but the stories assigned to them do.

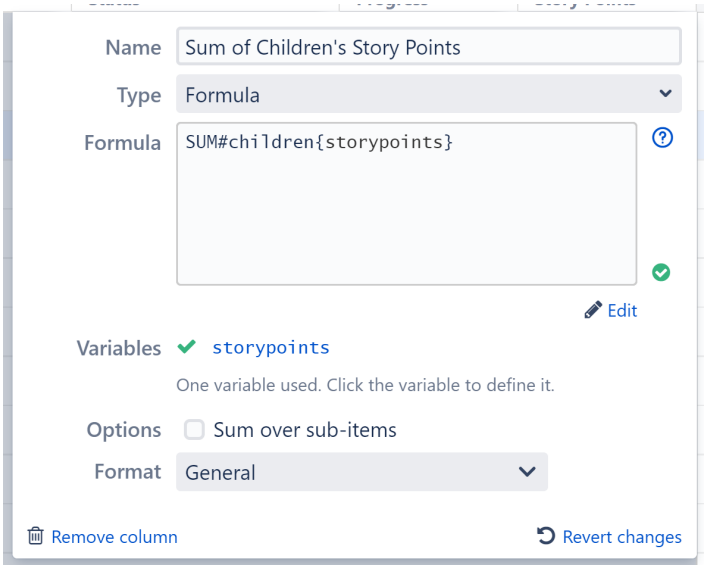


4.3.2.1 Step 1: Add a Formula Column to Calculated the Total Story Points

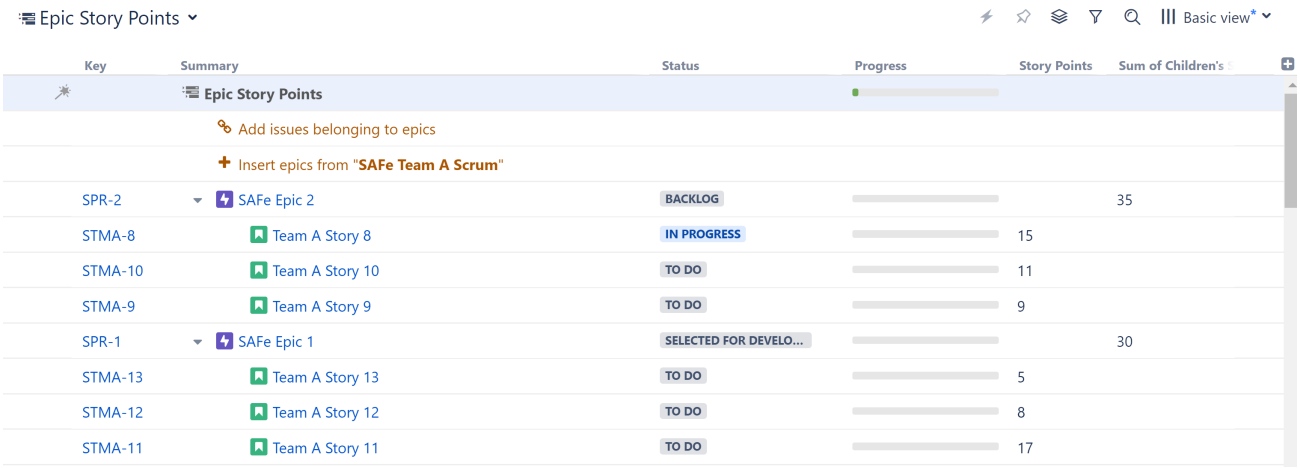
Click the **+** icon to the right of the column headers, and select **Formula...**



To calculate the total story points for each epic, we're going to use the aggregate function(see page 246) "SUM" with the "#children" modifier. Give the column an appropriate name and enter the following formula: `SUM#children{storypoints}`



Since we only have epics and stories in our structure, this will only create a value for our epics, because our stories don't have any children.



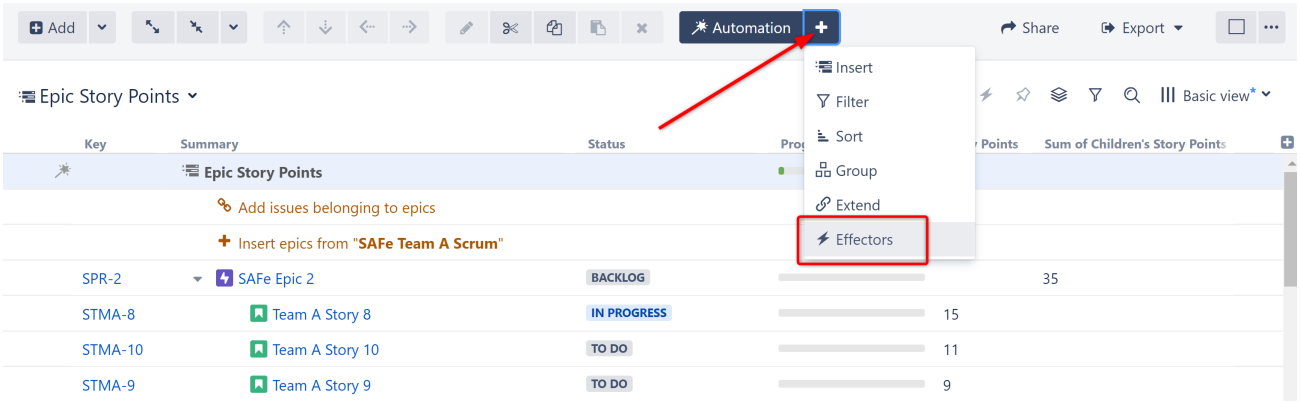
These aggregate values are only available in Structure, but we can use an Effector to take our Epic totals and write them to the Story Points field.

If you have an additional level below stories (such as sub-tasks), you will see Sum values for your stories as well - you can disregard those, because we're going to tell the Effector to ignore them when updating our Story Point values.

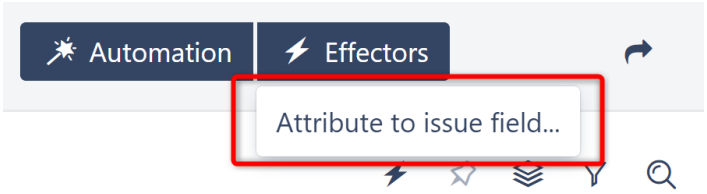
4.3.2.2 Step 2: Add an Effector

Now we need to take the information from the formula column and write it to the Story Points field for each Epic.

Open the Automation menu and select **Effectors**.

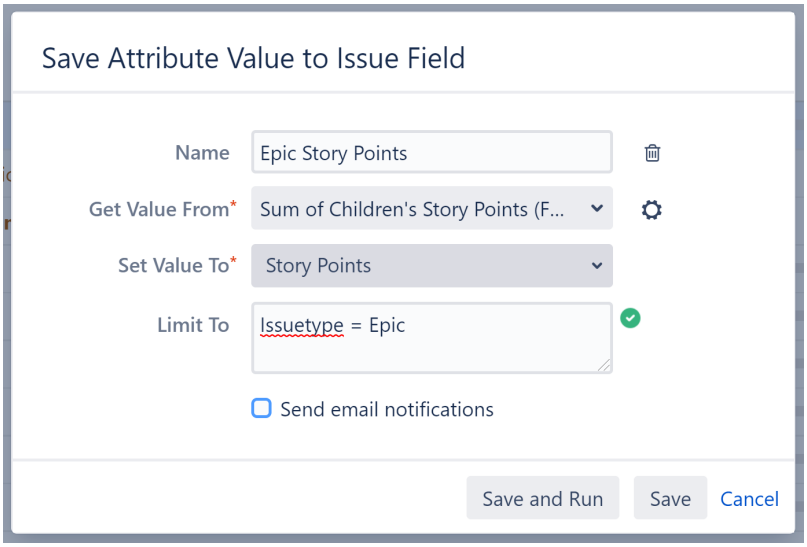


Choose **Attribute to issue field...**



On the Effector setting screen:

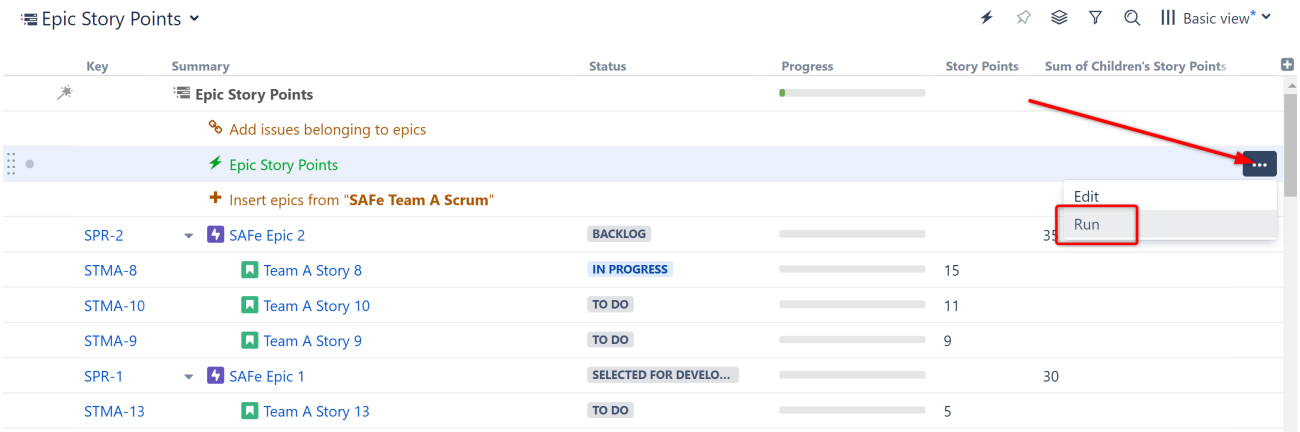
- The name field is updated automatically as you select the Effector properties. If you prefer, you can also click the edit button to enter a custom name.
- Get Value From: Select the formula column we added in Step 1
- Set Value To: Story Points
- Limit To: Since we only want to add Story Point values for our Epics, add the following JQL Filter: Issuetype = Epic (*Note: This is very important! If you don't set this filter, the Story Points data for stories will be overwritten by their children's sums, or removed completely if they have no children.*)
- Select whether email notifications should be sent when the Effector writes values to Jira



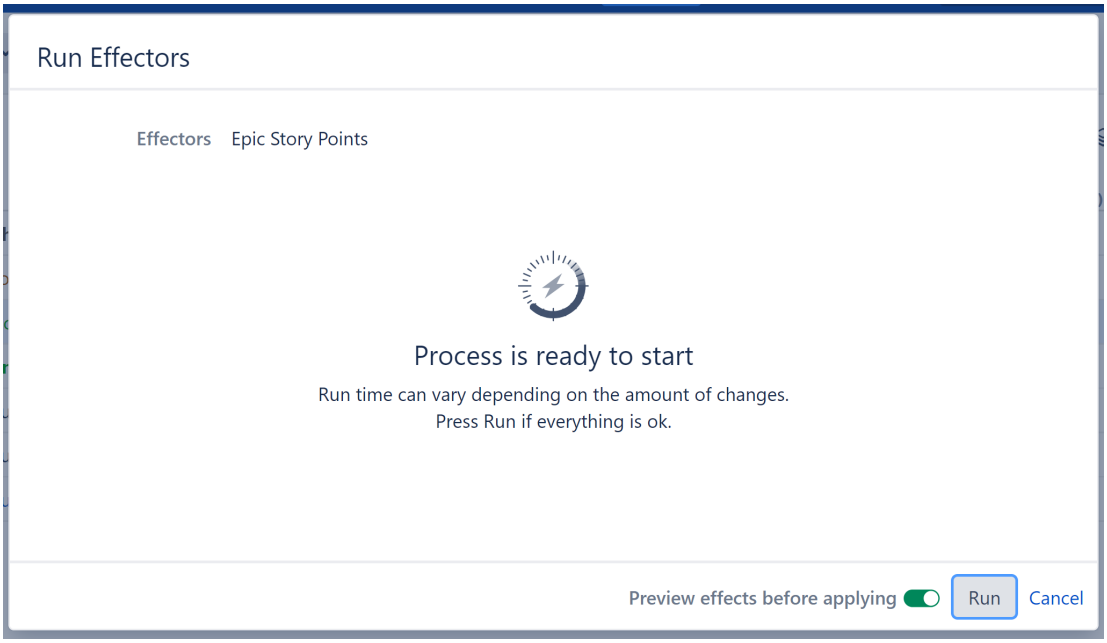
When you're finished, click **Save and Run** to run the Effector immediately, or click **Save** to simply add the Effector to the structure but not run it yet.

4.3.2.3 Step 3: Run the Effector

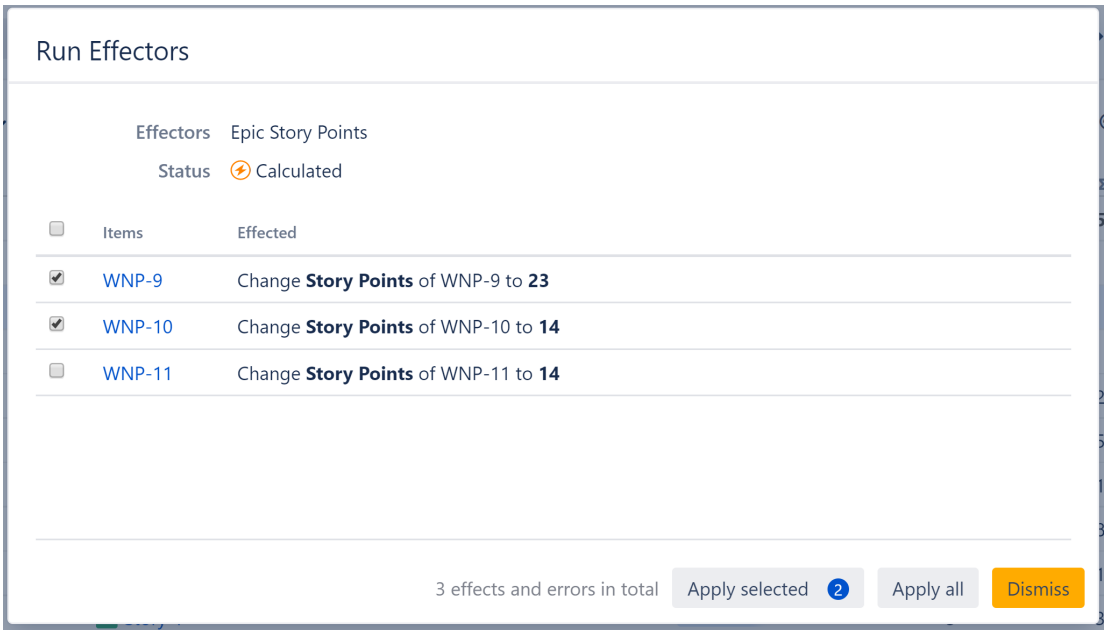
You can run an Effector directly from its settings screen (see above), or you can locate the Effector at the top of the structure and click the Action button (the three dots to the right of its row). Select **Run**.



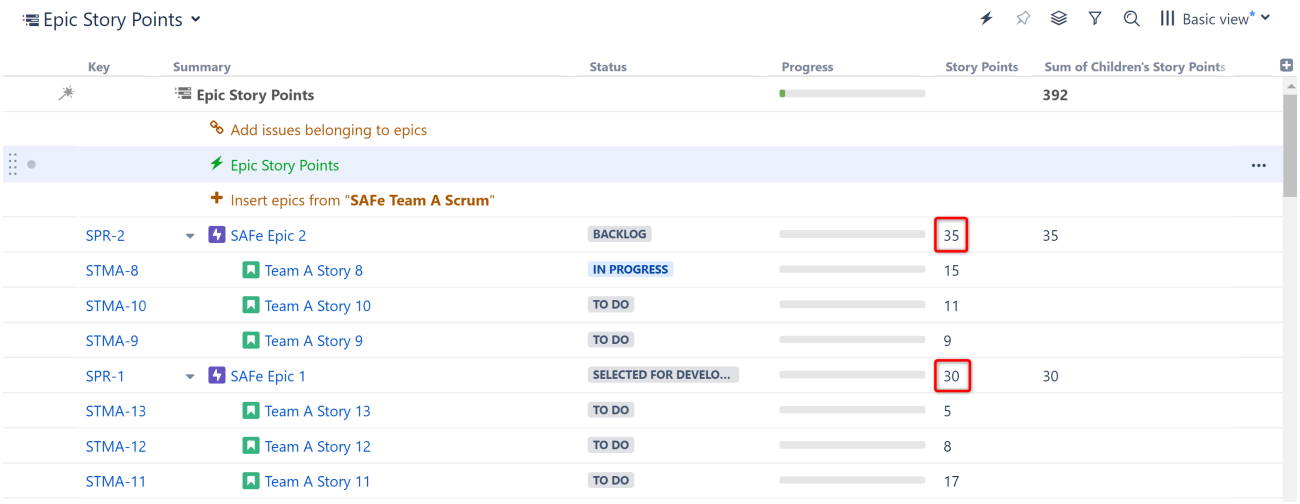
You have the option to **Preview effects before applying**. This allows you to view and approve every change that will be made by the Effector. Effectors update live Jira data, so we highly recommend using the preview option. Click **Run** to begin.



Once the preview is finished, you will see a list of changes that will be made by the Effector. You can select which changes you want made, or click **Apply All** to apply all changes.



Once you apply the selected changes, the calculated Story Points value will be written to each of our Epics:



4.4 Backlog Grooming for Teams

4.4.1 Situation

You want to perform backlog grooming for an entire team in one place.

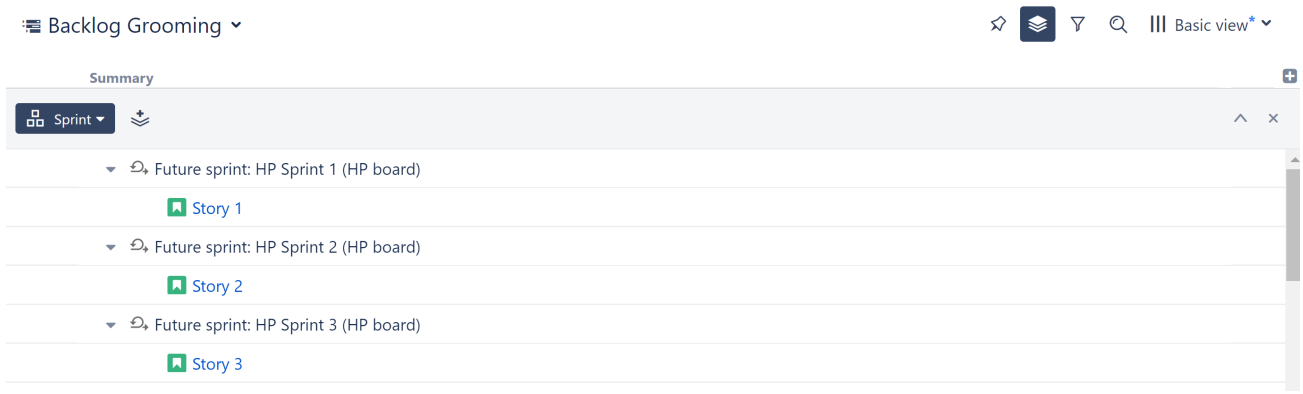
4.4.2 Solution

4.4.2.1 Step 1

Insert all the issues you need to groom. The easiest way to do this is using an Insert generator(see page 125) to add all issues from a project or board.

4.4.2.2 Step 2

Add a Group by Sprint transformation(see page 184).



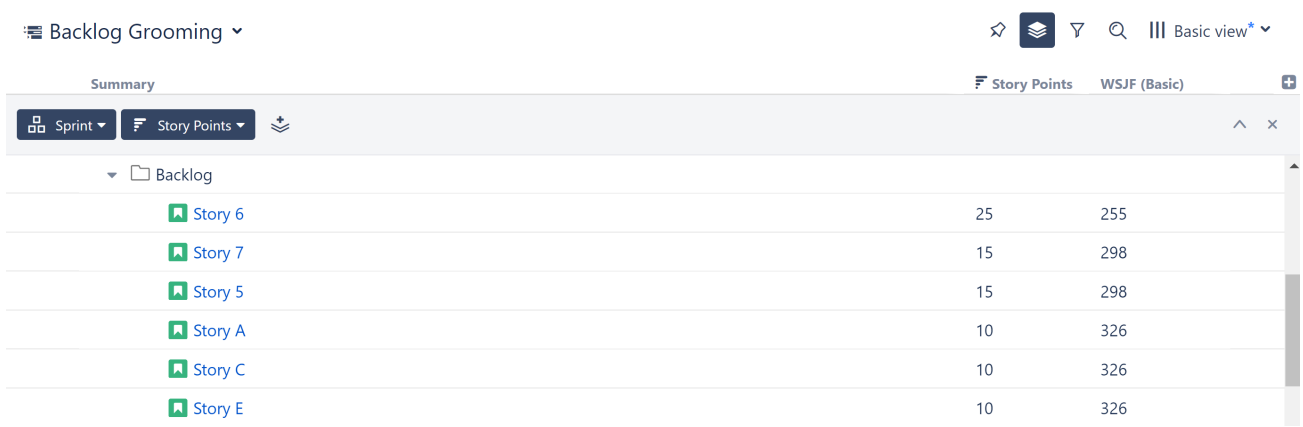
Sprints cannot be created in Structure. Make sure all the sprints you want to organize are already created, and at least one issue is assigned to each.

4.4.2.3 Step 3

Add columns to compare issue priority. We recommend adding:

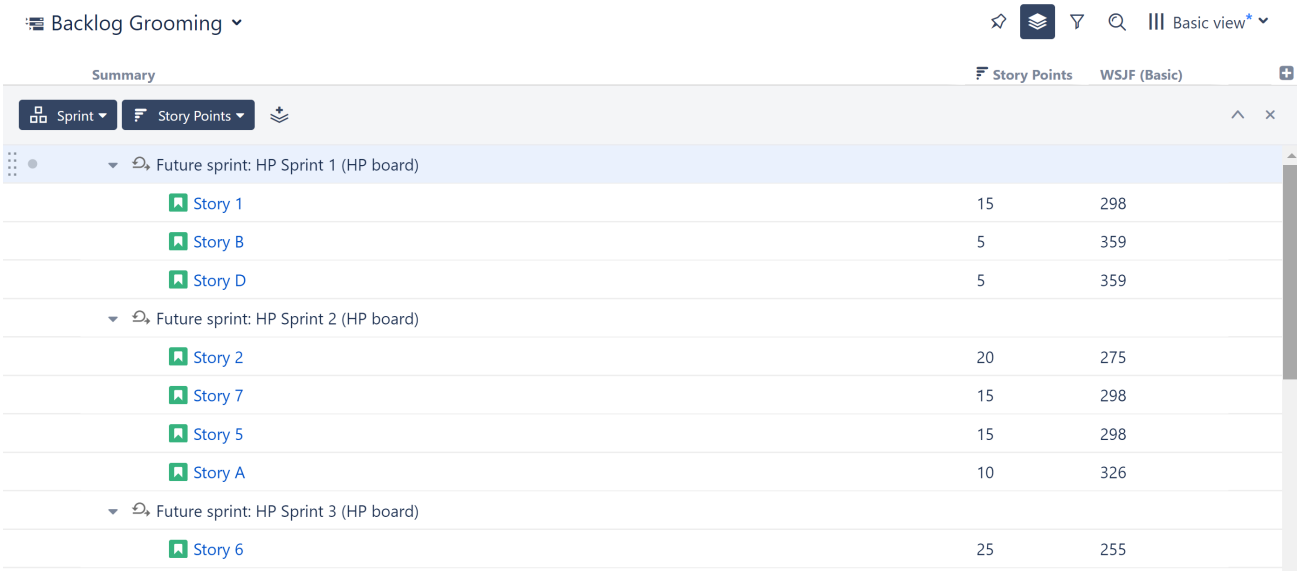
- Story Points (if you're using them)
- WSJF (or any other metric your team uses to prioritize issues or assess business value)

Click the header of any column to sort your issues by that value.



4.4.2.4 Step 4

Drag items to the appropriate sprint.



4.4.2.5 Optional Enhancements:

- Selecting sum over subitems on any numeric columns (Story Points, WSJF, etc.) allows you to see how many story points you've committed to for each sprint.
- Group by assignee under each sprint to see how the work is divided across your team.
- Add a Filter Sprints transformation to hide closed sprints.
- Use Structure.Gantt[64] to view any blocking dependencies between issues (to help prioritize them).

4.5 Correcting Inconsistent or Incorrect Link Types with the Extend Generator

4.5.1 Situation

You have several issues linked to a single initiative, epic, etc. You need to change the link type to be consistent with another project or team.

4.5.2 Solution

4.5.2.1 Step 1

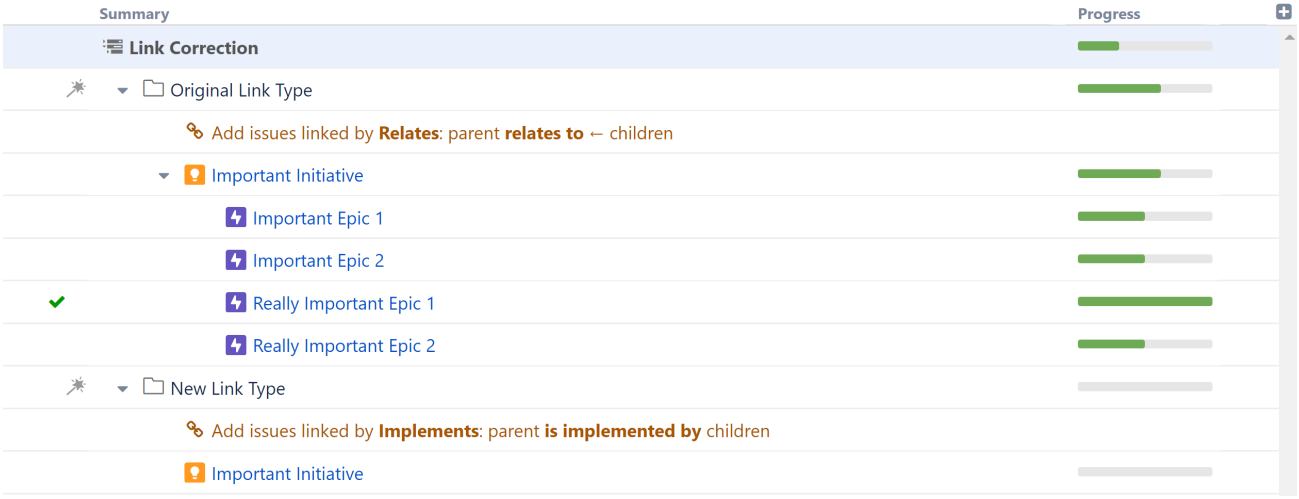
Create a new, blank structure and add 2 folders. **Add the parent initiative to both folders.**

64 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale

4.5.2.2 Step 2

Add an Extend with Issue Links generator(see page 147) to each folder:

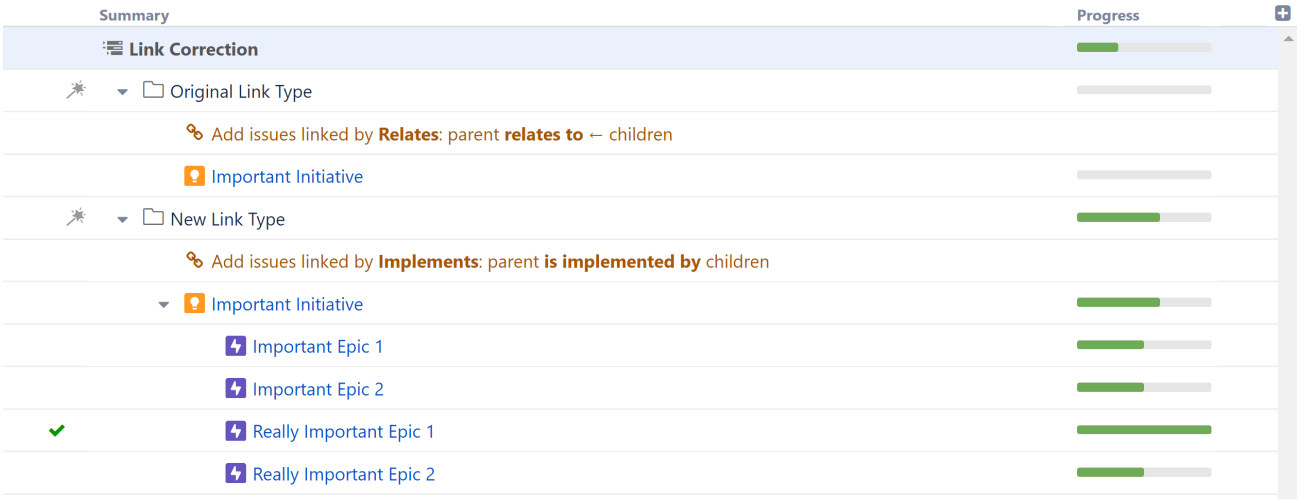
- First Folder - Link type = the current link type
- Second Folder - Link type = desired link type



(Make sure the **Allow changes via Structure** option is checked for both generators.)

4.5.2.3 Step 3

Select all issues under the generator added to the first folder, and **drag them to second folder**, as children of the initiative.



Structure will automatically replace all the old links with the new link type.

> Need to keep the original links too? When adding the original link type generator (to the first folder), uncheck **Allow changes via Structure**.

4.6 Resource Allocation

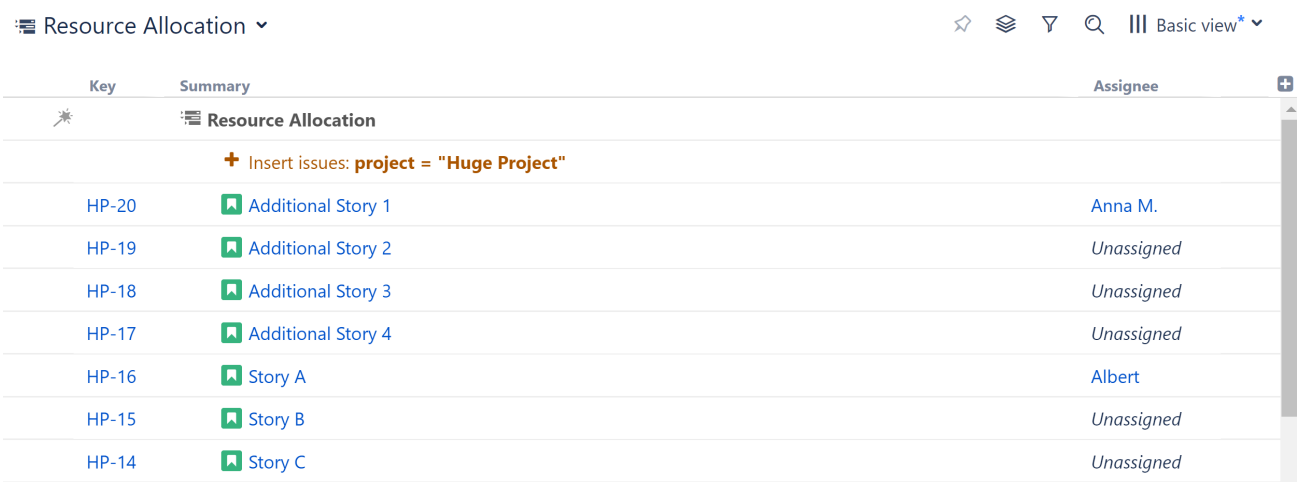
4.6.1 Situation

Team members are working on multiple projects, making it difficult to track how much work each individual is doing. Using Structure, you can quickly identify overallocation and underallocation, and reassign work on the fly.

4.6.2 Solution

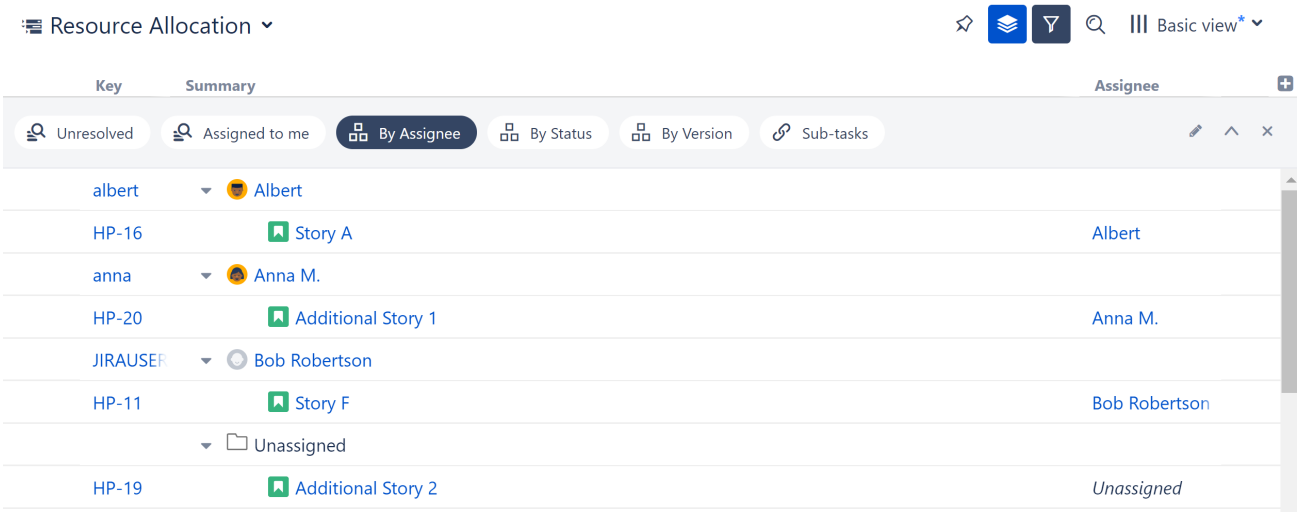
4.6.2.1 Step 1

Insert issues from every project the team members are working on. The easiest way to do this is using an Insert generator(see page 125) to add all issues from specific projects or boards.



4.6.2.2 Step 2

Open the Quick Transformations toolbar(see page 188) and select **By Assignee**, or add a Group by Assignee transformation(see page 184). (Hint: they both do the same thing!)



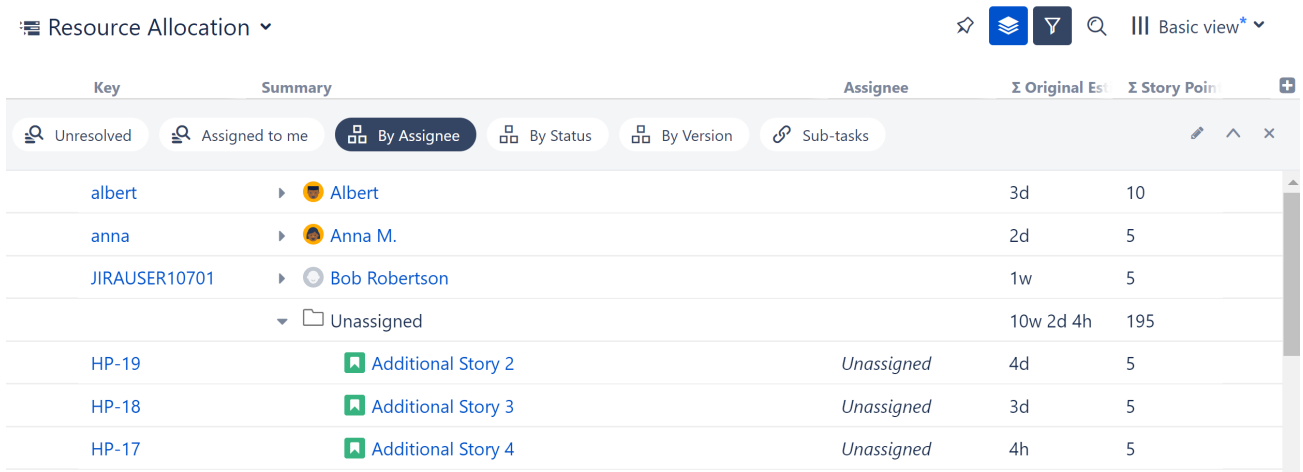
> ⚠ This will create a grouping for every team member with at least one issue in the current structure assigned to them. If someone doesn't have any issues assigned to them, they won't appear here. To fix this, simply assign an unassigned issue to that team member, and a new grouping will appear.

4.6.2.3 Step 3

Add columns to compare workloads. We recommend one or more of these:

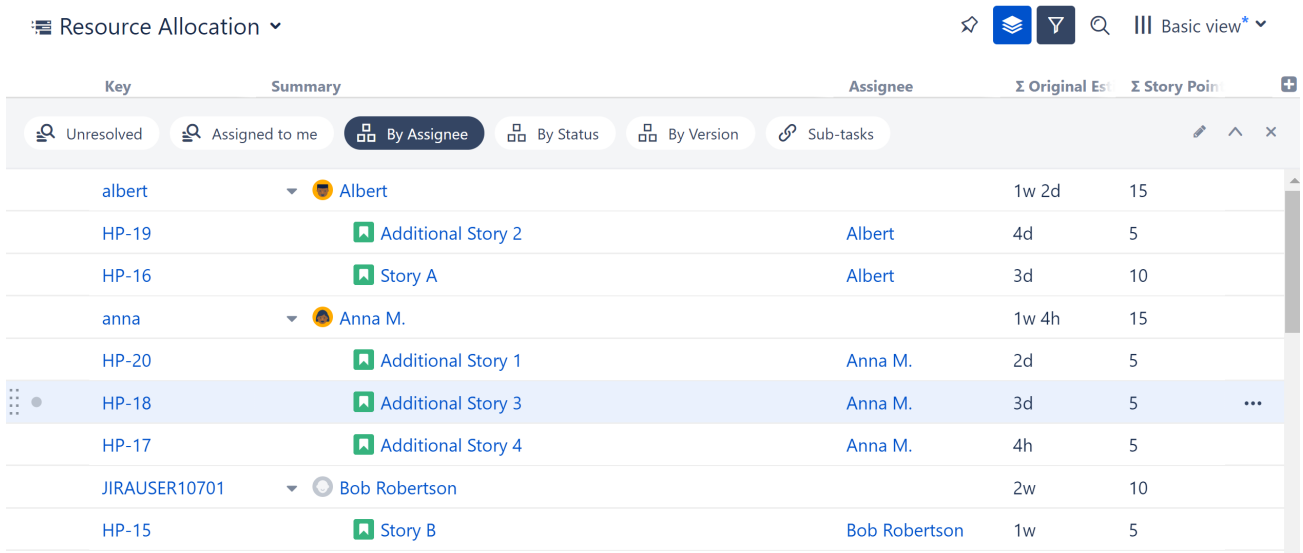
- ∑ Story Points (if you're using them)
- ∑ Original Estimate
- ∑ Remaining Estimate

Using the Totals columns, the values for each issue will be aggregated up to their assignee, so we can evaluate workloads at a glance.



4.6.2.4 Step 4

To assign/reassign tasks, simply drag issues from one assignee grouping to another.



4.6.2.5 Optional Enhancements:

- Limit the scope to a specific sprint by adding a Filter by Sprint transformation. (Haven't assigned sprints yet? Check out Backlog Grooming for Teams(see page 438)!)
- To allocate resources at the group level, group issues by a custom Team field instead of Assignee.
- Add a WSJF column to prioritize issues, and select Sum over sub-items to balance priorities across resources.
- Consider using Structure.Gantt[65] - it can identify and resolve overallocation automatically!

65 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale

4.7 Enable SAFe (Scaled Agile Framework) with Structure

The following are detailed instructions for visualizing SAFe in Structure.

4.7.1 Requirements

To do this, you will need:

- JIRA Software and Confluence
- Structure
- Structure.Pages
- Structure.Testy

Ready-to-use demo space: Live Demo[66]

4.7.2 Setup

We assume that your JIRA is prepared for scaling. The most common configuration is to create separate projects for each level:

- Kanban project for Portfolio level, with the following Epic issue type and statuses: Funnel, Reviewing, Analyzing, Portfolio Backlog, Implementing and Done
- Kanban project for Solution level, with the following Issue types: Capability and Enabler
- Kanban project for Program level, with the following issue types: Feature and Enabler

Features are named "Epics" in JIRA, so you can rename them in Administration. Projects for each Team (Scrum or kanban, Scrum is preferable) and Issue types (User Story, Bug, Improvement, Enabler) can be left as is.

4.7.3 Step-by-step guide

1. **Define your company Strategy in Confluence**
   Create a Space in Confluence for high-level documents and describe the Strategy of your company.

2. **Define Themes**
   Define Strategy more precisely by dividing it into several themes, each a sub-page of Strategy.

3. **Link Themes to Epics**
   Define Epics, or high-level tasks for implementing your theme. Link JIRA issues with their corresponding themes in Confluence.

4. **Add Capabilities**
   Since you have a separate project for Capabilities, divide your Epics into more precise Capability issues and put them into a Capability Board. Don't forget to link each Capability with a corresponding Epic! You can use standard JIRA issue link types, but it will be more convenient to create you own[67] "implements"/ "is implemented by" link type.

---

66 http://alm.works/safe-demo
67 https://confluence.atlassian.com/adminjiraserver071/configuring-issue-linking-802593137.html

5. **Add Features**
   Add details to any large Capabilities and divide them into Features. Don't forget to link each Feature to its corresponding Capability.

6. **Stories**
   Each Feature can be divided into more detailed Stories. Don't forget to link Features and Stories with an epic link.

7. **Sub-tasks**
   If you need more detailed tasks, divide Stories into Sub-tasks.

8. **Put it all together with Structure**
   Create an empty Structure and visualize everything in one place: http://recordit.co/lFYAzeBezV
   - Find your Strategy page and add it to the Structure: http://recordit.co/0vt6mNYsqR
   - Click the Automation button to enable automation. Then click the **+** icon next to the **Automation** button, and use the Extend generator to add the following rules:
     - Add all child pages (extend | child pages): http://recordit.co/5l5fqPM4c3
     - Add issues linked to pages
     - Add Capabilities linked to Epics
     - Add Features linked to Capabilities
     - Add Stories linked to Features
     - Add add Sub Tasks linked to Stories

9. **Take it to the next level!**
   Now that your structure is created, you can modify it to fit your needs:
   - You can Group issues on a specific level. For example, group by Sprint on the Team level to track progresses.
   - You can prioritize issues by sorting by WSJF.
   - You can specify your Objective during PI planning in Notes.

4.8 Creating an Advanced Formula Column

A number of interesting metrics would have a formula that involves sums of other formulas. For example, to calculate the % of bugs in a sub-tree, you need to divide the total number of bugs (calculated as a formula) by the total number of issues (also calculated as a formula).

There are a couple of ways you can accomplish this within a formula column(see page 197). You could use multiple formulas as variables(see page 205) or use columns as variables(see page 203). In the following example, we're going to recreate one of our bundled formulas(see page 219), BugFix %, using temporary columns as variables.

4.8.1 1. Write the Main Formula

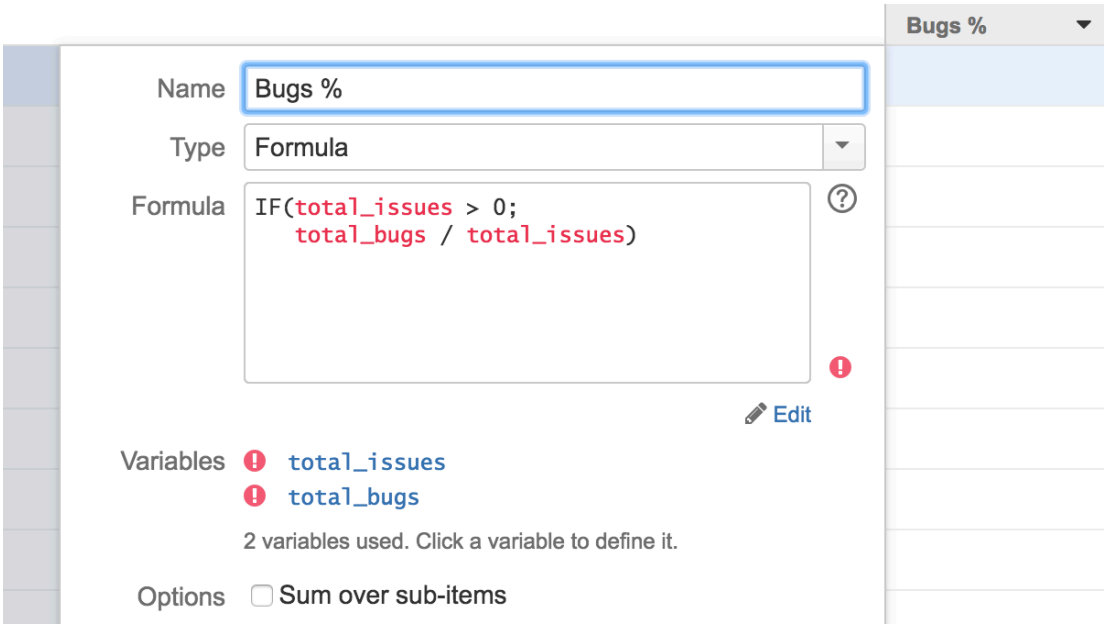
Create a new formula column and define the main formula. We're using **total_bugs** and **total_issues** as variable names, but they really could be anything.

In the **Formula** section, enter: `If(total_issues > 0; total_bugs / total_issues)`

A few things to note:

- We don't have total_bugs and total_issues defined at the moment, so they are marked red. That's OK.
- We use the `IF()` function to avoid dividing by zero.

- We don't use the Sum option, because this is a ratio. The summing is going to happen when total_bugs and total_issues are calculated.

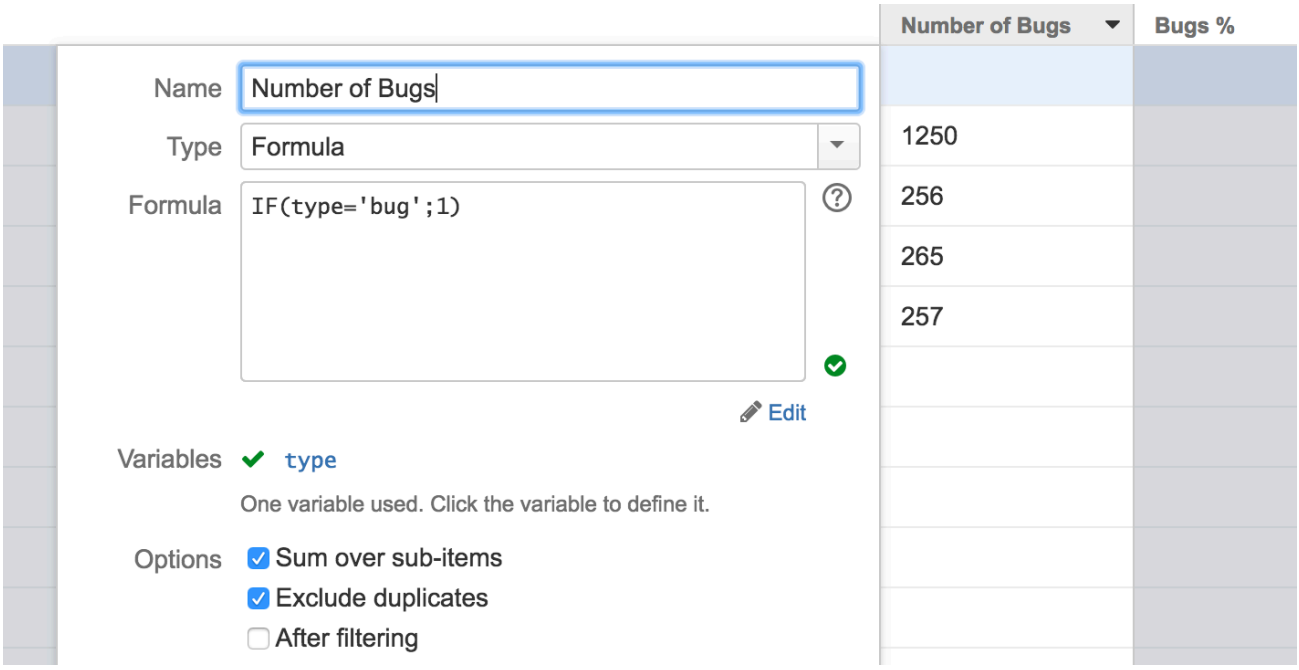


4.8.2 2. Create a Temporary Column with Total Number of Bugs

Next, we'll create a temporary column to calculate the total number of bugs.

In the **Formula** section, enter: `IF (type='bug';1)`

- For an individual row, we use the `IF()` function to check whether the issue type is Bug. If so, the formula returns a **1** for that row.
- The values are summed up (Sum over sub-items), with duplicate items excluded.
- The **type** variable is automatically mapped to the Issue Type attribute.

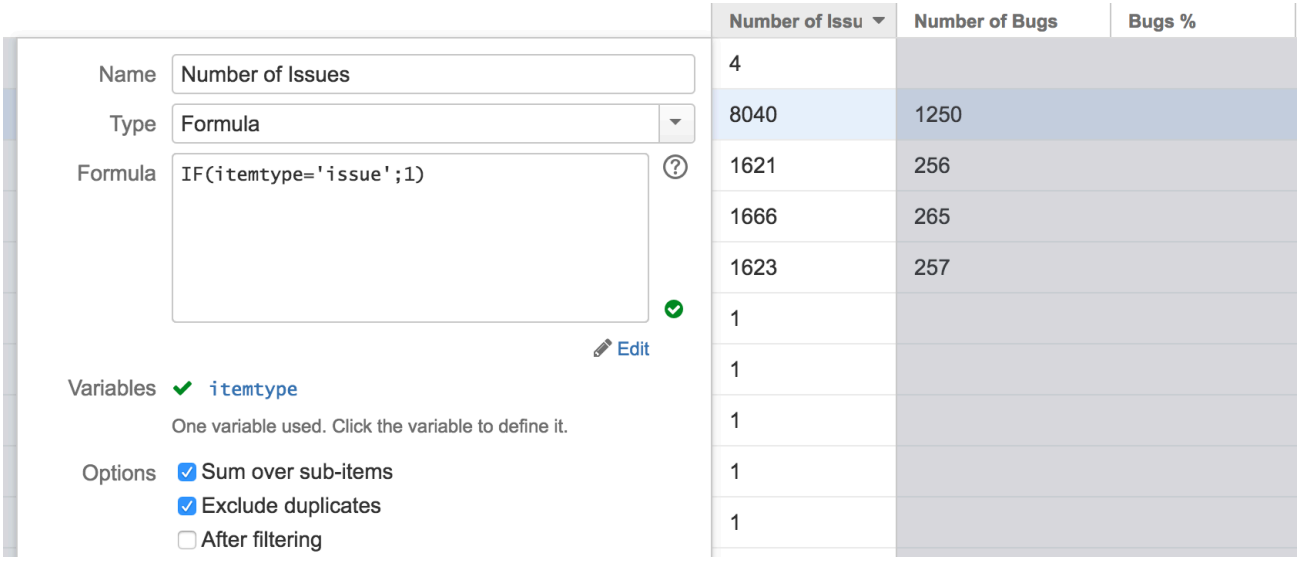


4.8.3 3. Create a Temporary Column with Total Number of Issues

This time, we'll create another temporary column to calculate the total number of issues.

In the **Formula** section, enter: `IF (itemtype='issue';1)`

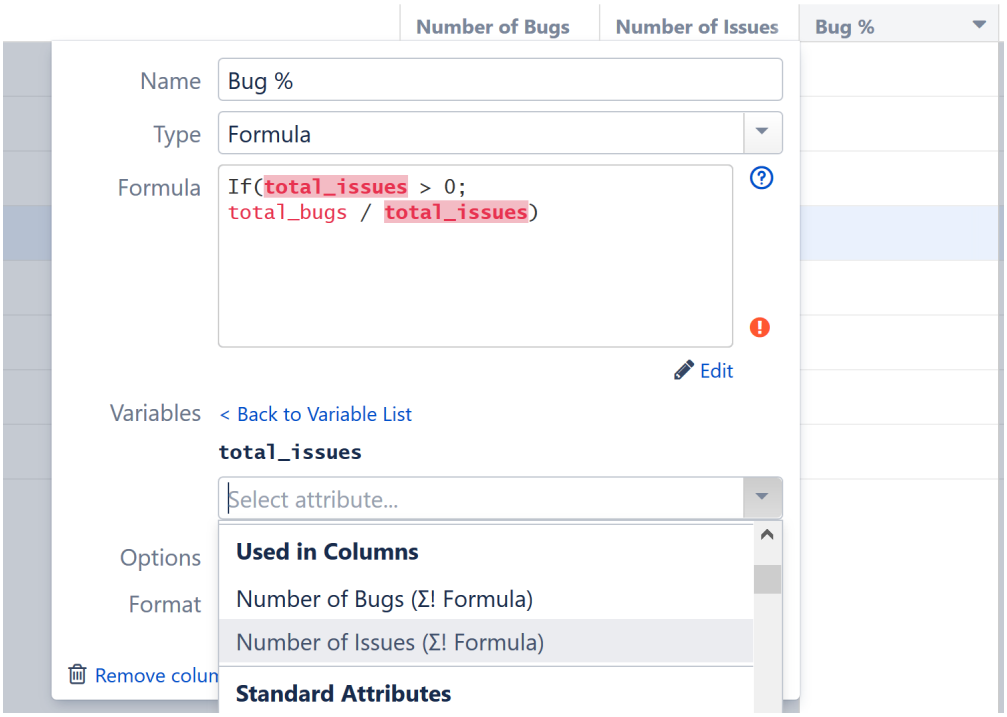
- Note that we still have to use `IF()`, because items in the sub-tree could be folders or other non-issue types.
- The values are again summed up (Sum over sub-items), with duplicate items excluded.



4.8.4 4. Define the Main Formula Variables

Now it's time to map the main formula's variables to the values calculated in the temporary columns:

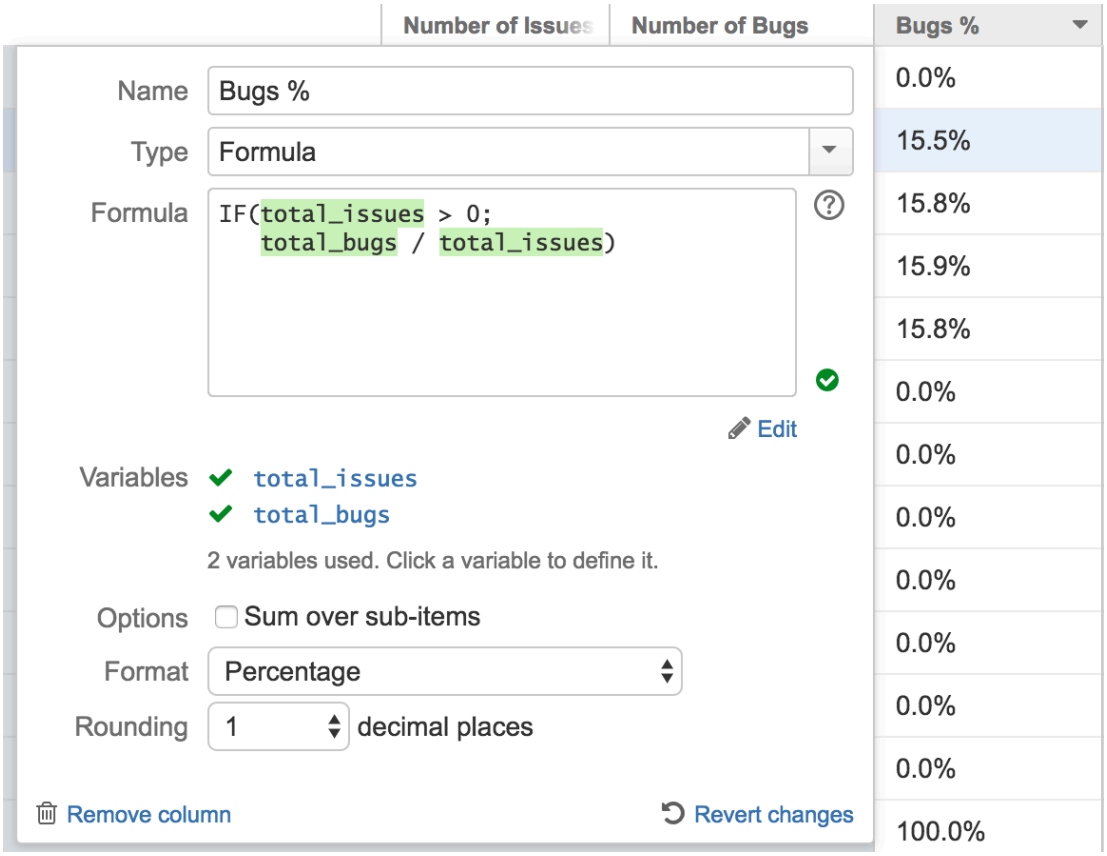
1. Go back to the main column configuration and click the **total_issues** variable.
2. In the attribute selector, scroll down to the **Used in Columns** section and select the "Number of Issues" entry.
3. Do the same to assign the **total_bugs** variable to the Number of Bugs entry.



> Please note that if you're selecting an attribute defined in another column (from the "**Used in Columns**" section of the attribute selection drop-down), you **copy** the attribute definition from another column at that moment, rather than creating a link to that column. That means that even if the other column is removed or if its content is changed, the formula configuration will keep using the attribute as it was defined at the moment you configured the variable.
>
> If you change the formula in the temporary column and want the change reflected in your main final formula, you will need to reassign the variable in the main formula.

4.8.5 5. Choose an Appropriate Format

Before we finish, we need to change the format to **Percentage**. And that's it - you've created an advanced formula!



As a final clean-up, you can remove the temporary columns from the grid. The Bugs % column has all the information it needs to do the calculation.

4.9 Track Overdue Tasks with Structure

4.9.1 Situation

You want to easily identify:

1. Overdue tasks
2. Number of overdue tasks per epic
3. Ratio of overdue tasks to total tasks for each epic

4.9.2 Solution

All of these can be accomplished by adding a few simple formula columns.

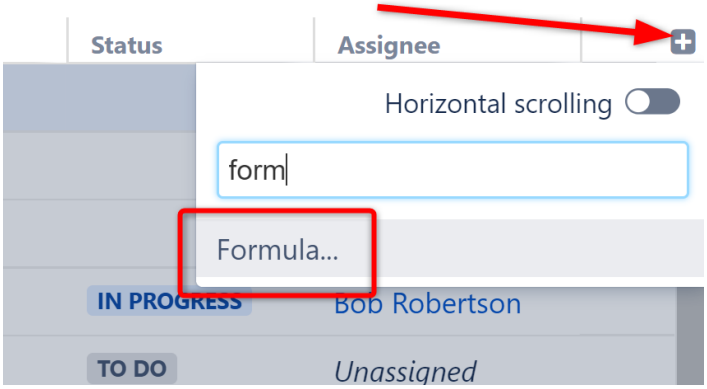
4.9.2.1 Step 1: Build Your Structure

Before creating any formula columns, we recommend organizing your structure into a hierarchy. While this isn't strictly necessary for identifying overdue tasks, it's extremely useful - and essential for aggregating overdue tasks at the epic level or higher.

If you're not familiar with how to build automated structures, we suggest that you start by reading Building a Structure with Automation(see page 50).

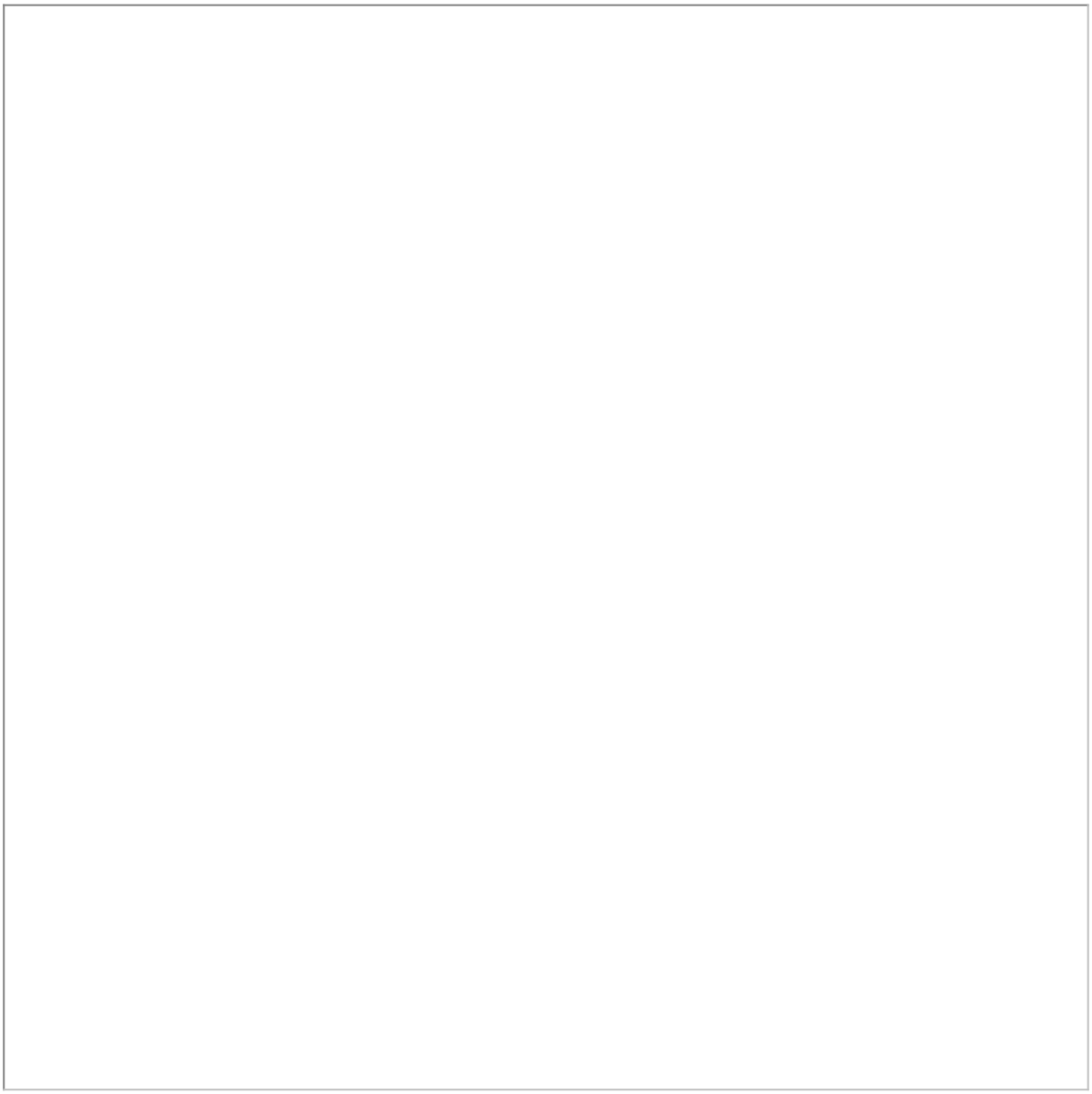
4.9.2.2 Step 2: Mark Overdue Tasks

Once you have your structure in place, the next step is to identify all the overdue items. To do this, we need to add a formula column(see page 197). Click the **+** button to the right of the column headers, and select **Formula.**



Give the column an appropriate name and enter the following into the Formula section: `if(due_date`[68] `< today(); "overdue")`

68 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6



If you used "due_date" in your formula, the variable will automatically be mapped to the Due Date attribute. If you used another name for that variable (for example, “due”), you will need to define the variable by pointing it to the appropriate Due Date attribute.

Variations

This is just one way to mark overdue items. You could also:

- Create a more visual warning that highlights not only overdue items, but also items coming due. You can read how to do this in Wiki Markup in Formula Columns(see page 207).

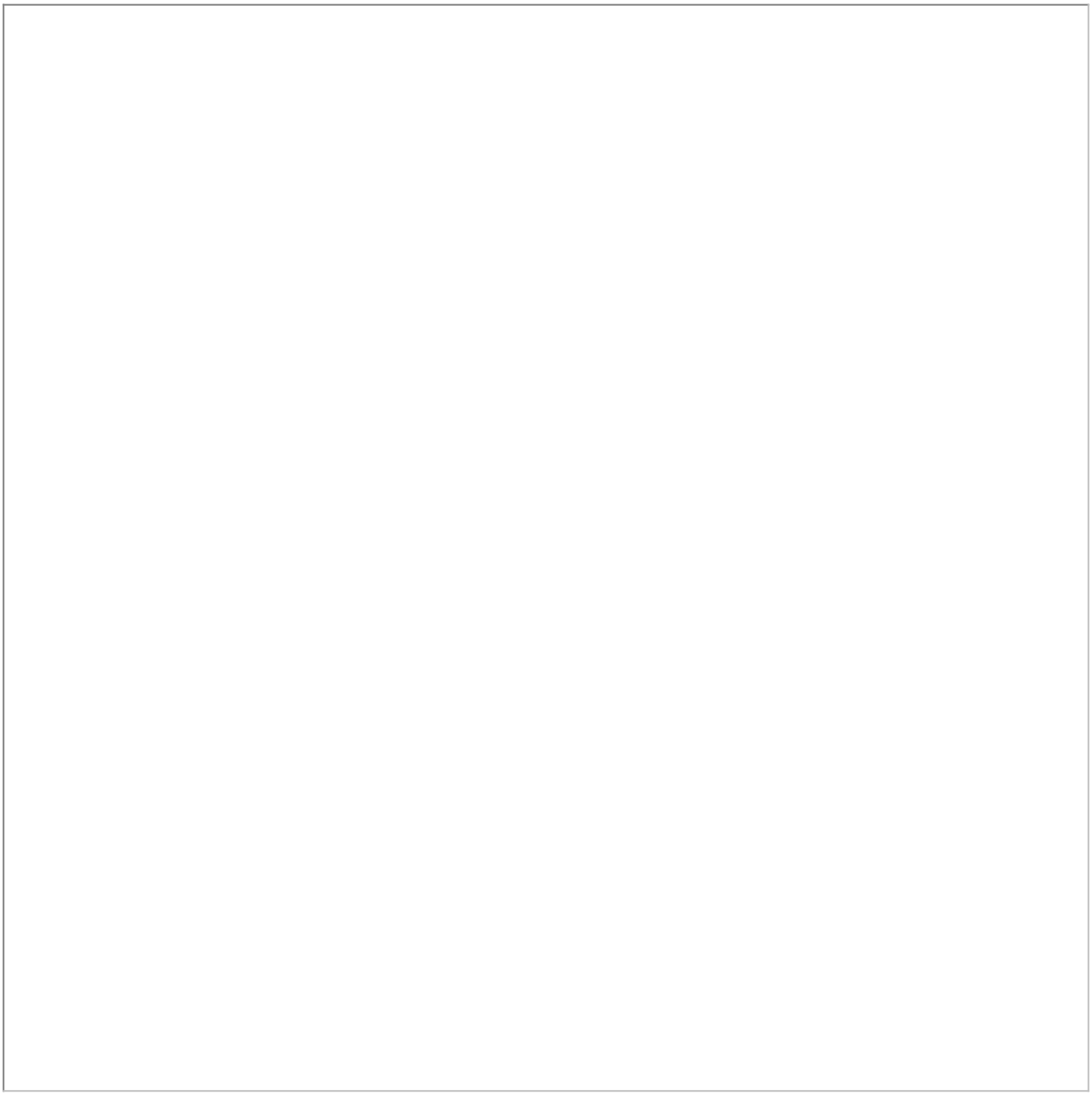
- Create a formula that relies on another due date. For example, you may want to flag issues that aren't completed by the due date of their parent epic. In this case, change the formula to: `if(type[69] = "Story" and status[70] != "Done" and parent{due_date[71]} < today(); "overdue")`
- Aggregate overdue items for epics (we'll cover this next!)

4.9.2.3 Step 2: Aggregate Overdue Items for Epics, Initiatives, etc.

In the previous example we were simply returning the string “overdue”, but if we give that "overdue" flag a numeric value ("1"), we can then use the Sum over sub-items option to aggregate those values up the hierarchy. To accomplish this:

1. In place of the formula we created in step 1, enter the following: `if(type[72] = "Story" and status[73] != "Done" and due_date[74] < today(); 1)`
2. Check the Sum over sub-items box

69 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
70 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
71 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
72 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
73 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
74 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6



Notice that in this example, we have several issue types above the epic level. The overdue numbers represent the number of overdue stories beneath each of these, aggregating at each level - so "Upgrade Website" has 3 overdue items, one from the stories under "Update Apache web server" and two from the stories under "Test website performance".

In this example, we've limited our scope to only stories that are not yet "Done" - depending on your situation, you may want to adjust this to include other issue types, add additional qualifications, etc.

We could have just as easily added a "Team" level here, to identify overdue items by team - or any other custom field we want to track by!

4.9.2.4 Step 4: Calculate the Ration of Overdue Tasks to All Tasks

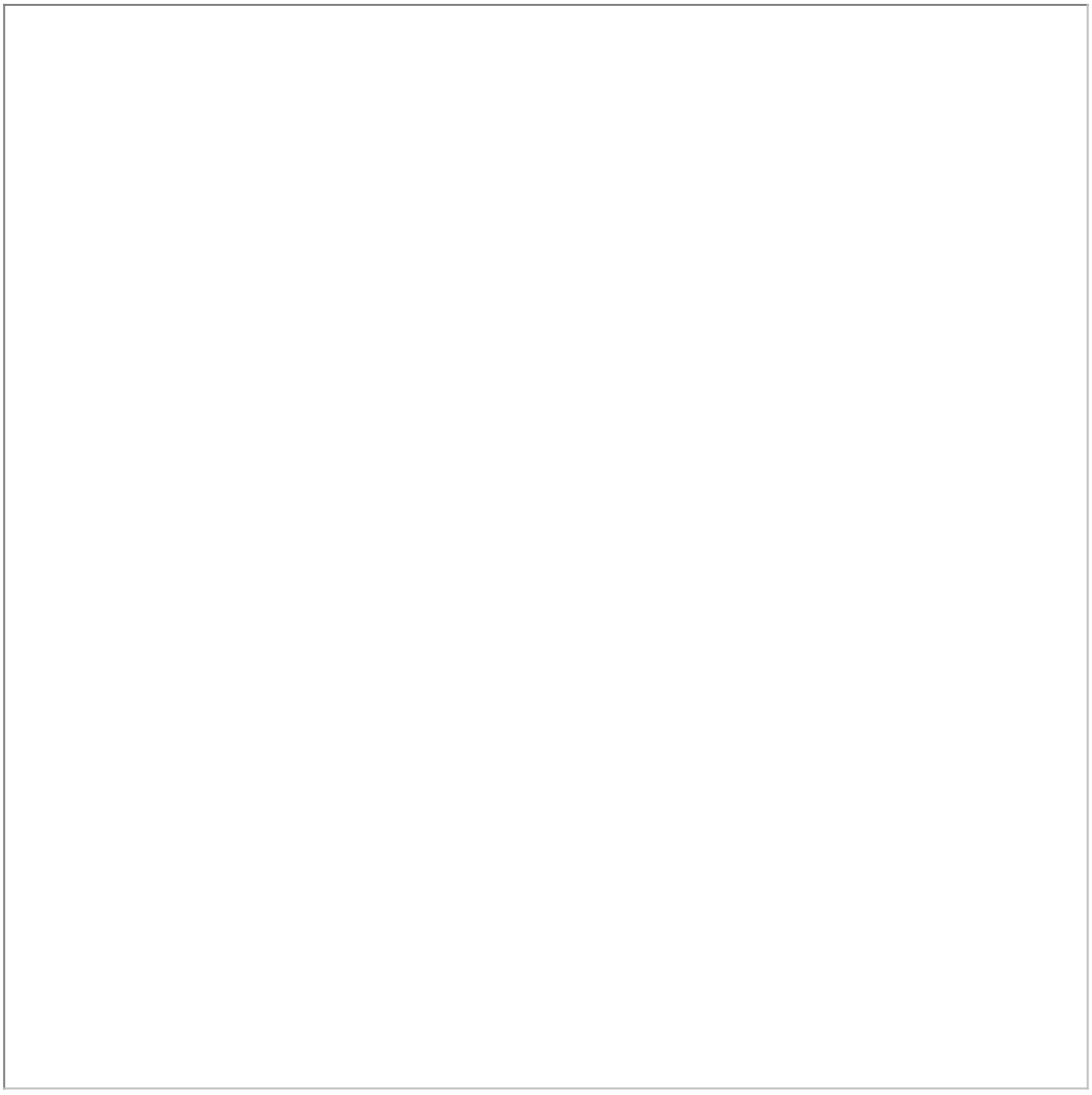
So far, we've just identified and counted our overdue stories. But this doesn't paint a complete picture. 1 overdue item out of 257 is very different from 1 out of 5!

To get a better understanding of how these overdue items relate to the big picture, next we're going to:

- Calculate total stories for each higher level in our hierarchy
- Calculate the ratio of overdue stories to total stories

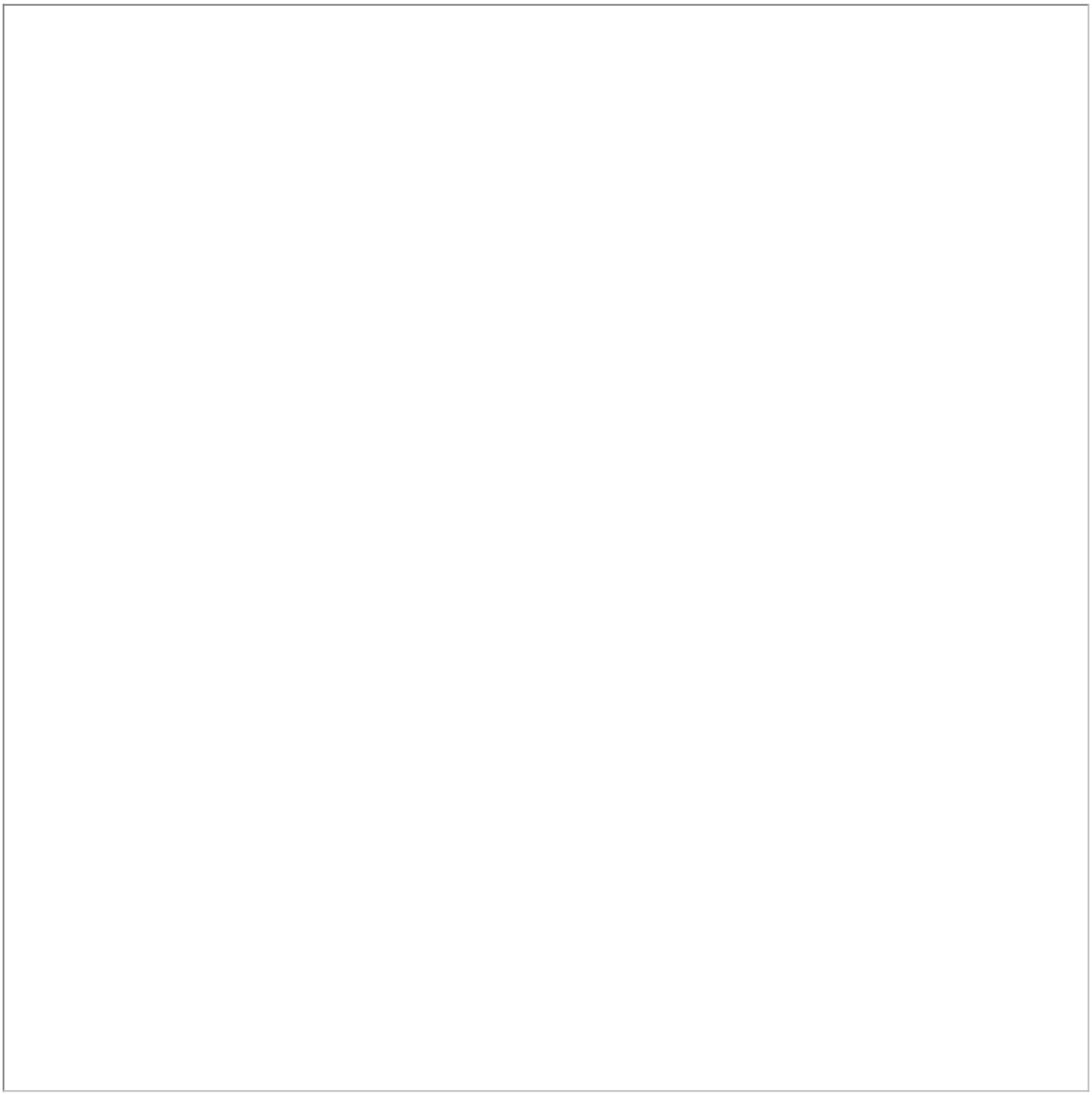
First, we need to create another Formula column, which we'll use to calculate our total stories. For this column:

- Enter the following formula: `if(type = "Story"; 1)`
- Check the Sum over sub-item box



Next, we'll create a third formula column to compare our values from the first two. For this column:

1. Enter the formula: `if(total_stories`[75]`; overdue`[76]` / total_stories`[77]`)`
2. Map your "total_stories" and "overdue" variables to the appropriate columns



For this formula, we wrapped the expression in an if statement so we wouldn't get a DIV/0 error on lines without a value for total stories.

75 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
76 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6
77 https://nick.demo.almworks.com/secure/StructureBoard.jspa?s=6

At this point, you can keep all three columns in your structure (we recommend saving them as a new view(see page 339)), or get rid of one or both of your original columns - our ratio column will keep their original calculations, even if those columns are deleted or changed. (This also means if you make changes to our Overdue or Total Stories column, you'll need to re-select those columns in the Variables section of your Overdue Ratio formula to apply those changes there.)

5 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(see page 467)
2. Setting Up Structure License(see page 473)
3. Getting Started with Structure(see page 458)

Search

Contents:

- Structure Best Practices(see page 459)
  - Monitoring and Troubleshooting Structure Usage(see page 466)
- Installing Structure(see page 467)
  - Migrating Data from Structure 2 to Structure 3(see page 468)
  - Memory Guidelines(see page 469)
  - Uninstalling and Reinstalling Structure(see page 471)
  - Upgrading and Downgrading(see page 471)
- Setting Up Structure License(see page 473)
  - Structure License Parameters(see page 476)
  - When Structure is Available for Free(see page 477)
  - License Maintenance and Expiration(see page 477)
- Selecting Structure-Enabled Projects(see page 477)
- Global Permissions(see page 478)
  - Who Has Access to the Structure(see page 478)
  - Restricting User Access to Structure(see page 478)
  - Changing Permission to Create New Structures(see page 479)
  - Changing Permission to Access Automation(see page 480)
  - Changing Permissions to Configure and Run Effectors(see page 480)
- Changing Structure Defaults(see page 481)
  - Changing Default Structure(see page 482)
  - Changing Default View Settings(see page 482)
  - Changing Default Options for the Issue and Project Pages(see page 483)
- Attribute Sensitivity Settings(see page 483)
- Structure Backup, Restore and Migration(see page 485)
  - Backing Up Structure(see page 486)
  - Restoring Structure from Backup(see page 487)
  - Migrating Structures(see page 488)
- Automatic Structure Maintenance(see page 490)
- Workflow Integration(see page 494)
- Running Structure on Jira Data Center(see page 496)
  - Archived Projects and Structure(see page 496)
- Anonymous Usage Statistics(see page 497)
  - Structure Usage Statistics(see page 498)
- Structure Files Location(see page 498)
- Turning Off Optional Features(see page 498)
- Advanced Configuration and Dark Features(see page 499)

- Structured Predictions(see page 507)
  - Configuring Structured Predictions(see page 508)
  - Using Structured Predictions(see page 510)
  - Structured Predictions FAQ(see page 513)
- Synchronization(see page 514)
  - Import Synchronization(see page 515)
  - Export Synchronization(see page 516)
  - Installing a Synchronizer(see page 517)
  - Modifying Synchronizer(see page 518)
  - Removing Synchronizer(see page 519)
  - Turning Synchronizer On and Off(see page 519)
  - Running Resync(see page 520)
  - Synchronization and Permissions(see page 521)
    - Changing Permission to Manage Synchronizers(see page 522)
  - Protection from Synchronizer Cycles(see page 522)
  - Bundled Synchronizers(see page 523)
    - Sub-Tasks Synchronizer(see page 524)
    - Filter Synchronizer(see page 524)
    - Links Synchronizer(see page 528)
    - Agile Synchronizer(see page 531)
    - Status Rollup Synchronizer(see page 534)
  - Undo Synchronizer Actions(see page 537)
  - Copying Synchronizers(see page 538)
- System Requirements(see page 539)
- Confluence Gadget - Admin Configuration(see page 541)
  - Adding Structure Gadget to Confluence Configuration(see page 541)
  - Setting Up CORS Filter in JIRA(see page 542)
    - Nginx Configuration Option(see page 542)
- Clearing Structure Caches(see page 543)
- Anonymizing Users(see page 544)
- Structure Troubleshooting(see page 545)
  - Collecting Support Zip(see page 546)
  - HAR Network Report(see page 546)
  - Troubleshooting Synchronizers(see page 547)
  - Structured JQL Troubleshooting(see page 549)
  - Collecting Performance Snapshots(see page 549)
    - Performance Snapshot Without Yourkit Plugin(see page 551)
  - Sending Files to Support Team(see page 555)
  - Troubleshooting Performance and Stability Issues(see page 556)
- Migrate to Cloud(see page 558)

5.1 Structure Best Practices

Because all organizations are different, and teams within those organizations are different still, Structure was designed to be versatile and flexible enough to handle a diverse range of unique use cases. Therefore, there is no single, best strategy to rolling Structure out across an organization. However, there are best practices which can be can be utilized to reduce disruption and maintain peak performance levels.

We will review these best practices, collected from our experience as well as customer feedback. It is important to understand that the best practices referenced here were developed with optimal system performance at the core.

⚠ Important information regarding Performance Considerations can be found below.

- Initial Considerations(see page 460)
- Basic Terms & Principles(see page 460)
- Things to Keep in Mind(see page 461)
- Common Approaches to Building(see page 462)
- Performance Considerations(see page 464)
- Additional Resources(see page 465)

5.1.1 Initial Considerations

5.1.1.1 Organizations New to Jira

If you have Structure as part of your Jira solution and your organization is still new to Jira, it may be best to consider the solution as a whole. That is to say, include it in your basic Jira onboarding process. A lot of our training material and resources are well suited to this. There are plenty of solutions consultants who can help with training as well.

5.1.1.2 Organizations Established in Jira

It is highly recommended to proceed with an incremental rollout in this situation. By utilizing Structure configurations, you can select specific groups and/or projects that will have access to Structure initially. You can then expand the access as your teams become more familiar. To add an additional layer of control to the roll-out, you can also limit the type of access each user or role has to read or edit structures. Typically, the first groups should be your champions - those who may have recommended Structure in the first place and whom you will rely on to guide additional users through best practices.

ⓘ For detailed instruction on how to adjust permissions, read Enabling Structure(see page 459)

5.1.2 Basic Terms & Principles

5.1.2.1 Automation

Automation uses generators, i.e. special rules, that tell a structure what issues to show you from Jira and where to place them within the structure. Whenever a structure is open, these generators are running and rebuilding your structure as changes are happening to the associated Jira data. In this way, you know that your structure is always up-to-date and relevant

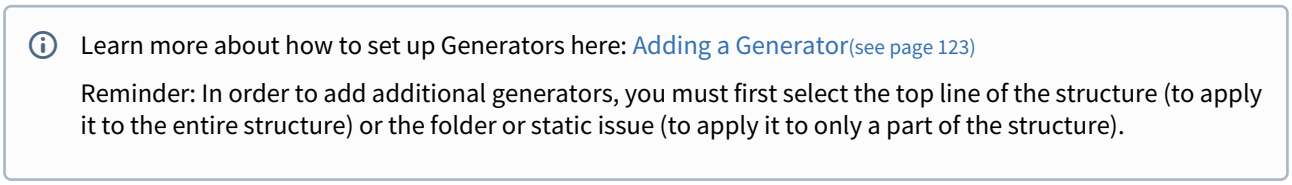
ⓘ To learn more about Automation a good how to video can be found here: Automation(see page 122)

Generators

- Insert - pulls in an initial set of issues, from which the rest of the structure is built
- Filter - hides some issues, or certain levels, from the structure
- Sort - reorders issues by the values in a specified field
- Group - organizes issues under the values of a field
- Extend - adds additional issues based on links to the issues already in the structure

Generators can only be added under a “static” part of the structure. In other words, anything that was not populated by another generator. Below are the two most common locations & their use case.

1. At the top of the structure - this will apply the generator to the entire structure. This is the best option if you need to organize the entire structure in the same way.
2. Under a folder or issue - this limits the generator to just the items beneath that folder or issue. This allows you to combine several sets of issues in one structure and organize them differently by adding different generators under each folder.



5.1.3 Things to Keep in Mind

Before we get started setting up a structure, there are a few key items to keep in mind as you get going. These are important to maintaining desirable performance levels.

1. Extend generators should have limited levels . The default is set from 1-10. We recommend adjusting for 1-2.
   Click here to read more...

   When added to a structure, an Extend generator checks every configured level in the structure to determine whether issues at that level need to be extended. For example, a Linked Items Extender will check every issue in every configured level to see whether or not a specific link type is present, and add linked issues whenever necessary. Checking an individual issue for links doesn't take long, but checking thousands of issues may.

   When configuring extenders, we recommend limiting the scope to just those levels that need to be extended. This will prevent your structure from checking unnecessary issues and can provide a significant performance improvement for large structures.

   See Generator Scope(see page 169) for more information

2. Use Filter generators sparingly . Adjust the initial insert generator query instead.
   Click here to read more...

   It’s not only the final number of issues in a structure that is important. It's also important to check the number of issues before any filter generators are added.

   For example, if you add an Insert generator with several Extend generators, they may pull 100k issues into the structure, but then a Filter generator may remove most of those, so you only see a few thousand issues. Even though it looks like a small structure, it still has to add all those original issues (and then put extra effort in to remove most of them), which could result in very poor performance

3. Aim for multiple, specialized, structures . Avoid large “all encompassing” structures.
   Click here to read more...

   Some structures are simply too large or too complex to be loaded as fast as desired. This may be fine for infrequently-used structures, but for structures that are used every day by multiple users, we recommend either decreasing the size of the structure (see above) or splitting the large structure into two or more smaller structures.

   For example, instead of having a global structure used by all teams, create several smaller structures specific to each team for day-to-day use, and reserve the larger global structure for cross-team planning sessions.

5.1.4 Common Approaches to Building

5.1.4.1 Top Down

We recommend utilizing this approach when a full overview is needed. Specifically where it is important to roll-up data such as estimates, time spent and progress from lower levels all the way to the top. It is strongly recommended to limit access to the appropriate users, such as project and program managers. We do not recommend using this method for a team's main structure for daily work or as the default structure.

Click here to read more...

Because of the nature of the use case, such structures can get pretty big. We are visualizing the entire scope of work. It's not necessarily a problem, but these structures are usually more performance heavy than smaller structures, especially if you have many levels connected by links. It's perfectly fine to use them when a full overview is needed, but it may not be the best approach for a general-use structure utilized by many users on a daily basis.

Additionally, this type of structure should not be used as a default structure. Default structures are opened by default on the issue page, which means every time a user opens an issue, this whole structure is loaded and then filtered to only show the part of it related to the issue.

How to build:

Start by adding top-level items using an Insert generator, and then use a number of Extend generators to pull in related items. Here are the common steps:

1. Add the Insert generator that will add the top level items. The JQL Inserter is the most flexible option, but it may be desirable to add issues from a particular Agile board.
2. Add Extend generators to pull in child issues. Depending on your setup, you may have to add several extenders to visualize different types of relationships. For example, you could build a 4-level structure as follows:
   - First level - add all Initiatives using a JQL Inserter
   - Second level - add epics under those initiatives with an Issue Link Extender
   - Third level - add stories using a Stories Under Epics Extender
   - Fourth level - add sub-tasks using the Sub-tasks Extender

> ⚠ It is strongly recommended to set the Levels option for all Extend generators - this will make everything more consistent and help with performance (especially in larger structures). In the example above, you'd set the Levels to Current level only for the first Link Extender. To pull in stories under epics, you will set the levels option: *2 to 2*, since the Epics are on the second level, and we don't need to check if there are any stories under stories. The sub-tasks extender would then be applied to levels *3 to 3*.

5.1.4.2 Focus on a Level

This approach is best for when you want to see how the scope of issues you are responsible for fits into the full hierarchy. For instance, you want to view the stories from your sprint and visualize which Initiatives and/or Epics they fall under, as well as the sub-tasks they are comprised of.

Click here to read more...

You may be wondering why this scenario would not call for the Top Down approach as well. You are, after all, trying to view a Parent > Child > Grandchild hierarchy. The difference here is in the focus. We are primarily interested in the Child, or middle level of the hierarchy. For performance reasons, we would not want to pull in all of the top level, Parent, issues, just to filter some or most of them out.

A better approach in this situation is to start by pulling in the scope you want to see and then use Group generators to show the levels above. This will produce a much smaller structure straight away, and Groupers are more efficient in terms of CPU usage than Extenders.

How to build:

Use an Insert generator to add the initial scope - the issues you are focused on - and then use Group generators to show the parents and Extend generators to show the children. Let's look at an example where we want to build a structure using an Initiative -> Epic -> Story -> Sub-task hierarchy, but only with stories from a particular sprint:

1. Use an Insert generator to add the initial scope. In this case, we would use the JQL Inserter to pull in issues from a certain sprint.
2. To add children (in this case sub-tasks), add a Sub-tasks Extend generator (don't forget to set the correct level settings - in this case, Current level only).
3. Next, add a Group by Epic generator to show the direct parents. The stories will be distributed between the epics they belong to. Any stories without an epic will end up in the "No Epic" folder. If the parents are connected to children with issue links instead of epic links, use *Group by Issue Links*, and select the link type and direction.
4. To include Initiatives above your epics, add a second Group generator. There are two important things you need to do here:
    a. As you add this second Group generator, make sure you select the "Consider other groups" option. Without this option, the generator will try to group not the level we've just created, but the issues that were added by the original Insert generator.
    b. Once the generator is added, reverse the order of these two Group generators in the generators list, so that the one you want to group by first is higher.
5. If there are more levels above - keep adding more Group generators, until you get the highest level you need.

5.1.4.3 Visualize the Entire Scope

When you are in a hurry, but also want to make sure you didn't leave anything behind, this approach will come in handy.

Click here to read more...

This is a very quick and easy solution. It can also be very resource intensive. We recommend utilizing it only in situations where you are limited on time and are looking to populate from a particular project, component, version, team, etc. In other words, the more specific the focus the better.

How to build:

Unlike the top-down approach we used above, where we started with just our top-level issues, in this case we want to start by inserting our entire scope of issues. Then we will use other generators to organize them into the hierarchy. Here is how to build it:

1. Use an Insert generator to pull in all the issues you want to see.
2. Add an Extend generator to pull in the child issues like we've done before. The important difference in this case is that as you add Extend generators, you will be getting duplicate issues - one instance of the child issue is added to the top level by the Insert generator, and then another is added by the Extend generator. To hide the duplicates, proceed to the next step.
3. Add the Remove Inserter/Extender Duplicates Filter generator - this will hide any issues added to the top level by the Insert generator if they were also added later by an Extend generator. Any issues that are not duplicated will remain on the top level.
4. The Extend generators we have added could pull in some issues which are outside of the original scope we defined. For example, epics in our project can have some stories from other projects too, which the Stories Under Epics Extender would pull in. If we only want to see issues from our original scope, we'll need to add a JQL Filter generator with the same query we used originally.

It is very important to set level parameters for extend generators in this configuration - even more so than in the top-down approach. Failing to do so can seriously affect performance.

If you do not restrict the scope of Extend generators by setting correct levels, you will end up with not just 2 instances of all child issues, but with an instance per level, which can be a lot of issues if you have 4-5 levels.

5.1.5 Performance Considerations

5.1.5.1 Automation

When evaluating Automation performance, there are several factors that come into play

1. The size of structures (the number of issues in each structure).
2. The complexity and the depth of structures. The more levels a structure has, the more calculations need to be done to build it.
3. Number of concurrent users editing large structures.
4. The frequency of usage. If you have a really large structure that is used on a daily basis or is set as the default structure(see page 357), this may have a negative effect on the server performance. On the other hand, a very large structure that is only used by a small group of managers when they really need it would have minimal impact.

The number of structures with generators does not affect the performance - if a structure is not opened, its generators are not running.

To reduce the risk of Automation affecting Jira performance, we have introduced the Automation Timeout feature. This feature stops generation if it exceeds a certain, user-defined time period. The user is then able to adjust the generator settings. See Paused Generation(see page 172) for more details.

5.1.5.2 Progress, Totals and Formula Columns

The Jira fields used in columns, by themselves, have little impact on performance. They are loaded only for the structure elements which are currently displayed, in addition to a number of items below and above (to ensure smoother scrolling).

However, for progress, totals and some formulas, the values are loaded for the entire hierarchy. This is due to the fact that such columns aggregate data across the entire hierarchy. For structures with thousands of issues, this can become noticeable. This is still far less of an impact when compared to loading the hierarchy itself.

5.1.5.3 Structure on Issue Page

Users can see structures on the main Structure Board, on Jira dashboards, the project page and on the issue page. A structure which is visible in any of those locations requires generation of the structure.

For high traffic projects, this could affect performance significantly. It is recommended to use smaller and simpler structures as your default structures. To learn more about selecting the default structure for the Issue Page, see Structure Options for the Issue Page(see page 88).

You can also remove the Structure widget from the Issue Page(see page 498) completely.

5.1.5.4 S-JQL

S-JQL(see page 266) is an extension for JQL that allows you to run structure-based queries. Every time such a query is executed, the structure that is used in the query will be loaded. If you create a saved filter with S-JQL for a large structure and use it heavily somewhere, performance may begin to suffer.

5.1.5.5 Monitoring & Troubleshooting Structure Usage

The best way to check statistics on Structure usage, see structure sizes and assess load time is by reviewing the Performance Audit Log (PAL). For detailed information on the type of data which can be found in the PAL, as well as how to interpret it, additional information can be found here: Monitoring and Troubleshooting Structure Usage(see page 466)

5.1.6 Additional Resources

- Getting Started with Structure(see page 49)
- Structure User's Guide(see page 73)
- Structure Administrator's Guide(see page 458)
- Structure Developer's Guide(see page 842)
- Frequently Asked Questions(see page 561)
- Structure Troubleshooting(see page 545)

5.1.7 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.

5.1.7.1 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(see page 175) 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(see page 466).
- The number of rows vs unique items. If the number of rows is much bigger than the number of unique items, it means there are multiple duplicates in this structure. Sometimes this is necessary, but in many cases it indicates the structure was built incorrectly or inefficiently. See Building Efficient Structures With Automation(see page 466).

- 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(see page 266) 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.

5.2 Installing Structure

Structure is installed like most other apps.

1. Before installing Structure in production, make sure your Jira meets the Memory Guidelines(see page 469).
2. Open Manage add-ons, search for "Structure" by ALM Works on the Atlassian Marketplace and install from there.

   Alternatively, you can download the plugin JAR manually from the download page(see page 978) and either place it into the *plugins/installed-plugins* subdirectory under your Jira home (then restart Jira) or use the "Upload Add-on" link in Manage add-ons.

3. Press the **Get Started** button to finish the installation by installing a license key(see page 473).

Congratulations! You can now spread the word and help users get started with Structure – see Getting Started with Structure(see page 467).

If Structure Remains Disabled

It is possible that after you install Structure or enable it from Manage add-ons, the add-on will remain disabled. An error may or may not be shown. If you refresh the Manage add-ons page within 5-10 seconds and Structure is still disabled, see Structure Won't Start(see page 565) for possible causes and solutions.

5.2.1 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.

5.2.1.1 Creating a Backup of Structure 2.x Data

- If you still have Structure 2.x installed, create a backup of the current Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the entire `structure/` sub-directory under Jira home while Structure is disabled. See Backing Up Structure(see page 486) 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.

5.2.1.2 Restoring Structure Data from 2.x Backup

1. Use the **Administration | Structure | Restore Structure** menu and use any Structure 2.x backup made earlier. Note that it should be placed in the `import/` directory on your server.
2. If you used the "Export Structure 2.x Data" menu, you will be taken to the restore automatically.

5.2.1.3 After Data Migration

Upgrade Testy

If you have Structure.Testy installed, download and install the latest version of Structure.Testy, compatible with Structure.

Upgrading "Global Structure"

If you're using a "Global Structure" structure, which was created by default in Structure 2.x, you need to make sure that there's an "owner" of that structure. Otherwise, Automation(see page 122) will not work there.

1. Open **Structure | Manage Structure**.
2. Find Global Structure and check if it has an Owner assigned.

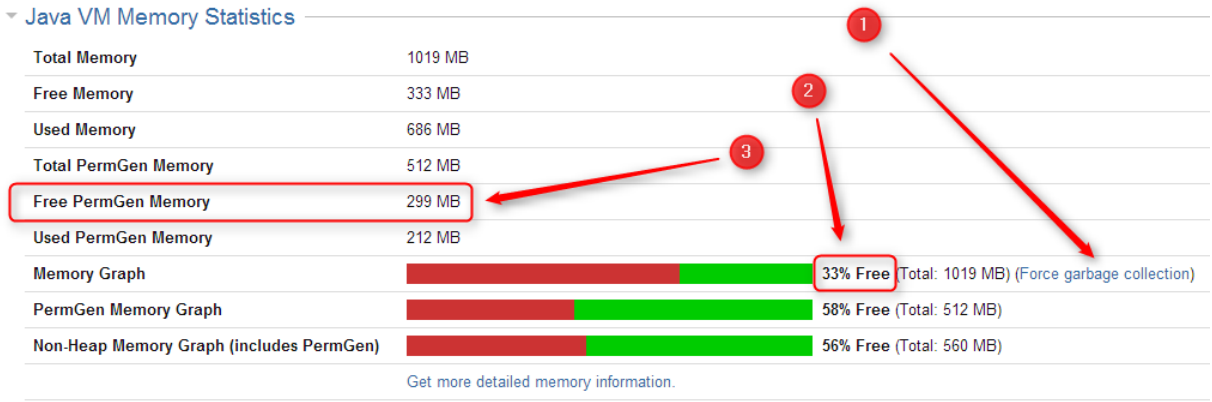
3. If it doesn't have an owner, click **Configure** and set yourself as the owner.

5.2.2 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).

5.2.2.1 Assessing Available Memory

1. Open menu **Administration | System | System Info** and scroll down to **Java VM Memory Statistics**.
2. Click **Force Garbage Collection**
3. Note the free % number of the **Memory Graph** (heap memory).
4. Note the absolute amount of **Free PermGen Memory** (non-heap memory for Java classes).



Memory Statistic	Recommended Value	Parameter in `setenv.sh` / `setenv.bat`
% of Free Heap Memory	25% – 50%	`JVM_MAXIMUM_MEMORY`
Free PermGen Memory (prior to Java 8)	100 – 200 MB	`JIRA_MAX_PERM_SIZE`

If you run Jira on Java 8, PermGen memory is not a factor.

All recommendations are for a general case and do not guarantee that you won't get `OutOfMemoryError`. Individual cases may vary.

5.2.2.2 Heap Memory Requirements

It is recommended that % of free heap memory is from 25% to 50%.

Structure requires about an additional 100 MB of heap memory. You can take your current statistic of **Used Memory** and **Total Memory**, add 100 MB to the **Used Memory** and calculate the recommended value for the **Total Memory**.

If you already have the recommended % of free memory, you can just increase total heap memory by 200 MB.

5.2.2.3 PermGen Memory Requirements

This section applies to Jira running on Sun/Oracle Hotspot Java VM only.

PermGen space is used for Java classes and may be depleted if you uninstall, install or upgrade plugins frequently, or if you don't restart Jira over a long period of time. Due to technical reasons, PermGen space might not get cleaned up from the obsolete classes, and you may end up with an `OutOfMemoryError: PermGen space` error.

Structure classes use only about 10 MB of PermGen space. But for the reasons just mentioned, it is good to have a safety margin with a free PermGen space of at least 100 MB.

5.2.2.4 Changing Memory Parameters

To change memory parameters, edit `setenv.sh` (on Windows, `setenv.bat`).

- To change the maximum amount of Heap space, edit the `JVM_MAXIMUM_MEMORY` parameter near the top of the script.

```
JVM_MAXIMUM_MEMORY="2000m"
```

- To change the maximum amount of PermGen space, edit the `JIRA_MAX_PERM_SIZE=256m` line. Alternatively, you can add the `MaxPermSize` parameter to `JVM_SUPPORT_RECOMMENDED_ARGS`. For example:

```
JVM_SUPPORT_RECOMMENDED_ARGS="-XX:MaxPermSize=400m"
```

You need to restart Jira for these settings to take effect.

5.2.2.5 Use 64-Bit Java

It is imperative to use 64-bit Java when allocating a large amount of memory to it (1 GB and more). To check if you're running 64-bit Java, look up the **Java VM** parameter on the System Info page.

5.2.2.6 Physical Memory Requirements

Avoid swapping at all costs!

The amount of physical memory should be enough to accommodate the whole heap and non-heap memory. If you have other Java or memory-intensive applications running on the same host, they all should fit in physical memory, plus you need to reserve at least 1 GB for the operating system, services, and file cache.

Do not allocate more memory to Jira if it cannot fit into physical memory! If Java running Jira starts swapping actively used memory, it will severely impact performance.

Sample calculations for a host running Jira and Confluence, with Apache and MySQL:

Jira	Heap: 2 GB Non-heap: 500 MB
Confluence	Heap: 2 GB Non-heap: 500 MB
Operating system Apache HTTPD MySQL	1 GB
Free memory margin / File buffers	2 GB
Total Physical Memory Required	**8 GB**

5.2.3 Uninstalling and Reinstalling Structure

5.2.3.1 Uninstalling Structure

You can uninstall Structure from the Add-on Manager the same way you uninstall other add-ons. You can also manually remove the Structure JAR from the `plugins/installed-plugins` directory when Jira is not running.

When you uninstall the Structure add-on, Structure data is **not removed**. It remains in the Jira database.

5.2.3.2 Reinstalling Structure

It is perfectly safe to uninstall Structure, then install it again. (This happens, for example, when you upgrade to a newer version.)

All Structure data will be there unless you manually remove it.

5.2.4 Upgrading and Downgrading

5.2.4.1 Upgrading

To upgrade Structure 3.0 or later:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)

2. Back up Structure data. Use **Administration | Structure | Backup Structure**. See Backing Up Structure(see page 486) for details. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Check Structure extensions. If you are using Structure.Testy, Structure.Pages, Structure.Gantt or other extensions, they may become disabled. In this case, you need to either upgrade them too (it might be a compatibility requirement) or enable them manually in the Add-on Manager. If they fail to enable, reinstall them (uninstall and install again).
5. Check plugins that integrate with Structure. As with extensions, make sure they are enabled and upgrade/reinstall as necessary.
6. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

For more specific instructions, please check the Release Notes(see page 572) for the version to which you wish to upgrade.

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

5.2.4.2 Downgrading

Reverting Structure to an older version is not always possible, because newer versions can modify the database, making it incompatible with older versions.

Simplified Downgrade

A simple downgrade is possible if the database schema hasn't changed. Check the Release Notes(see page 572) for the version you are downgrading from and look for a downgrade advisory. Proceed only if you have indications that it is safe to downgrade to the specific version you have in mind.

1. Uninstall Structure. This step is required, because Add-on Manager will not install an earlier version over a later version.
2. Install the version that you need.
3. Check Structure extensions and integrating add-ons. See the steps in the Upgrading section above.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors. **This is especially important with this kind of downgrade, because some errors may be subtle and not visible to users!**

Reliable Downgrade

Reliable downgrade requires a Structure backup file and manual access to the database.

1. Create a Structure backup using **Administration | Structure | Backup Structure**.
    a. Backup files are backward / forward compatible along Structure 5.x series. To downgrade to an earlier version, see Downgrading to Structure 3.0 - 4.6(see page 473). 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.

5.3 Setting Up Structure License

Unless your Jira runs on one of the free licenses(see page 477), 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.

5.3.1 Setting Up Evaluation License

1. Navigate to **Administration | Structure | License Details**.
2. Look at the **Current License** section - if there's no license there or if the license is expired, you will need to get an evaluation license or purchase a commercial license.
    - If the **Current License** section says you have a **Free License**, then your Jira must be qualifying for automatic free license and no further action is needed from you. See When Structure is Available for Free(see page 477).
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[78] 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.

78 http://almworks.com/structure/evaluate.html

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.

5.3.2 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	ALM Works website[79]	Atlassian Marketplace[80]
Managed at	The license key is sent to you by email	Manage with other Atlassian licenses at my.atlassian.com[81]

License key looks like this:	`-----BEGIN CERTIFICATE-----`
	
	`MIIEYTCCAkmgAwIBAgIGAT2oPFqOMA0GCSqGSIb3DQE...`
	` ... at least 20 lines of symbols ...`
	` -----END CERTIFICATE-----`
		`AAABEA0ODAoPeNp9UE1Pg0AUvO+v2MSbCc0uQZO...`
		` ... at least 4 lines of symbols ...`

Installation Instructions	1. If you have a license from Atlassian installed, first remove it in **Manage Add-ons**. 2. Open **Administration	Structure	License Details**. 3. Copy and paste the key to the **Install License** section and click **Install License**.	1. Open **Manage Add-ons**. 2. Locate and open the Structure section. 3. Copy the license key into the **License** box and click **Update**.
Uninstallation Instructions	1. Open **Administration	Structure	License Details**. 2. See the details of the installed license and click **Uninstall**.	1. Open **Manage Add-ons**. 2. Locate and open the Structure section. 3. Clear the license key from the **License** box and click **Update**.

79 http://almworks.com/structure/purchase.html
80 http://almworks.com/structure/marketplace.html
81 https://my.atlassian.com

	License from ALM Works	License from Atlassian
Purchasing differences	• Besides advance payments with credit card, wire transfer or other payment methods supported by our payment processor, we can also accept purchase orders on Net 30 terms. • VAT and taxes may be handled differently from Atlassian, as our payment processors are located in the USA and Germany. ALM Works is based in the US, and for direct purchases using Wire Transfer, we do not charge VAT or any other taxes.	• Purchasing from Atlassian is not available in certain countries[82].

5.3.3 Purchasing a Commercial License

Structure licenses can be purchased from ALM Works, from Atlassian, or through Atlassian Solution Partners and resellers.

5.3.3.1 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**.

5.3.3.2 Purchasing from Atlassian

You can purchase a license via Atlassian on the Atlassian Marketplace[83].

After the purchase is completed, the license key will be available on https://my.atlassian.com.

5.3.3.3 Purchasing from Resellers or Atlassian Experts

You can purchase through a reseller of your choice. Atlassian Solution Partners[84] 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.

82 http://www.atlassian.com/licensing/marketplace#generalmarketplacequestions-11
83 http://almworks.com/structure/marketplace.html
84 https://www.atlassian.com/partners

5.3.4 Migrating Licenses

You can convert a license of one kind into a license of another kind. Please contact sales@almworks.com[85] for assistance.

Next: Select which projects are enabled for Structure(see page 477)

5.3.5 Structure License Parameters

The following parameters are displayed in the **Current License** section when you install a Structure license.

Parameter	Meaning
License Type	Commercial, Evaluation or other
Licensee	Organization authorized to use the license
Serial Number	A unique number assigned to the license
Expires	If present, the license is not perpetual: it will expire at the specified date. After that date passes, Structure will not be available unless the license key is changed.
Maintenance Expires	If present, the license key can only work with the versions of Structure released prior to the specified date. If you need to use a newer version of Structure, you will need to renew maintenance.
User Limit	This is the maximum number of users allowed by Jira that are supported by this license key. The license that Jira runs on must allow this number of users or fewer.
Server ID	Although not shown in the license table, most licenses are tied to a specific Jira server ID and will not install on a server with a different ID. If you need to move a license key to a different server, please contact support.

85 mailto:sales@almworks.com

5.3.6 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.

5.3.7 License Maintenance and Expiration

5.3.7.1 Commercial License

Your commercial license for Structure (including Starter licenses) typically has no expiration date, so it can be used forever. However, it has *Maintenance Expiration Date* which limits which versions of Structure can be used with that license – you can only use the versions released prior to the expiration date.

To use versions released later, you need to purchase a maintenance renewal, which extends your maintenance expiration date by one year.

5.3.7.2 Evaluation License

Evaluation and temporary licenses have an expiration date. After that date, Structure will only be accessible in read-only mode (see below).

Make sure you renew the evaluation or get another license key prior to the expiration.

5.3.7.3 Read-only Mode

If the current license becomes invalid (for example, it has expired or you upgraded to a version of Structure that is not covered by the current license's maintenance), Structure will function in read-only mode.

Users will be able to view structures, but they won't be able to make any changes until a valid license is installed.

5.4 Selecting Structure-Enabled Projects

Structure can be enabled for any selection of Jira projects, or for none of them.

> ⚠ By default, Structure is enabled for all projects. To limit users' exposure to Structure, pick specific projects to be enabled for Structure.

To select which projects are enabled for Structure:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Enable/Disable Structure in Projects**.
3. Select whether Structure should be available for **all projects** or for **selected projects**.

4. In the latter case, update the **Selected Projects** list by selecting one or more projects and using the **Enable** and **Disable** buttons.
5. Click **Apply** when done.
6. If you disable projects that are already used in a structure (a structure contains issues from that project), you will be given a warning. You can opt to **Proceed with Changes** or cancel.
    a. If you proceed and disable a project that has issues in some structures, those structures will appear to the users without those issues.
    b. If you later enable that project again, the issues will reappear where they were (all structure changes taken into account).

Which projects are enabled for Structure affects Who Has Access to the Structure(see page 478)

5.5 Global Permissions

- Who Has Access to the Structure(see page 478)
- Restricting User Access to Structure(see page 478)
- Changing Permission to Create New Structures(see page 479)
- Changing Permission to Access Automation(see page 480)
- Changing Permissions to Configure and Run Effectors(see page 480)

5.5.1 Who Has Access to the Structure

Structure is visible only to specific users. Only users who have access to Structure will see the *Structure* menu and other user interface elements provided by Structure.

A user has access to Structure if all of the following conditions are met:

- The user has **Browse** permission on at least one of the projects that are enabled for Structure(see page 477).
- Structure is enabled for this user(see page 478):
    - 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.

5.5.2 Restricting User Access to Structure

By default, Structure is accessible to anyone who has *Browse* permission on structure-enabled projects(see page 477). You can further restrict this access level to one or more user groups.

To select who can use Structure:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Select Structure Users**.
3. Select whether Structure should be available to **Everyone** or to **Users in selected groups/roles**.
4. In the latter case, change the **selected groups/roles** list by selecting the second radio button and use the **Add Group/Role** section to add one or more required user groups or project roles. To set up required property, use the drop-down selectors to choose either **Group** or **Project** option, then choose the required group name or project/role combination and press the **Add** button to add it to the list. If **project** is set to

"Any", this means that the user should be in the specified role for any of structure-enabled projects(see page 477).

5. You can remove the permission option by clicking the trashcan icon on the right of the option.
6. Click **Apply** when done or **Cancel** to dismiss your changes.

Which projects are enabled for the Structure also affects Who Has Access to the Structure(see page 478).

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(see page 362)).

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.

5.5.3 Changing Permission to Create New Structures

By default, any logged-in user with access to Structure(see page 478) can create new structures of their own. However, you can restrict this ability to one or more user groups.

To select who can create new structures:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Select Who Can Create Structures**.
3. Select whether new structures can be created by **Anyone with access to Structure** or by **Users in selected groups/roles**.
4. If permissions are based on groups/roles, use drop-down menu to choose either **Group** or **Project**, and then select the required group name or project/role combination. To search the list, simply select the field and begin typing. Click **Add** to include the selected group/role. *Note:* If **project** is set to "Any", this means that users should be in a specified role for any of structure-enabled projects(see page 477).
5. You can remove permission option by clicking the trash can icon on the right of the option.
6. Click **Apply** when done or **Cancel** to dismiss your changes.

The user also needs general access to Structure(see page 478) 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(see page 362)).

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.

5.5.4 Changing Permission to Access Automation

By default, any user with Edit Generators access level(see page 362) for a structure can add and configure generators(see page 122). You can restrict this ability based on user groups and/or project roles.

To select who can edit generators:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Select Who Can Access Automation**.
3. Select whether generators can be changed by **Anyone with Edit Generators permission** or by **Users in selected groups/roles**.
4. In the latter case, change the **selected groups/roles** list by selecting the second radio button and use the **Add Group/Role** section to add one or more required user groups or project roles:
    a. Use the drop-down selectors to choose either the **Group** or **Project** option.
    b. Choose the required group name or project/role combination. (*Note:* If **project** is set to "Any", this means the user should be in the specified role for any of the structure-enabled projects(see page 477).)
    c. Press the **Add** button to add it to the list.
5. You can remove a permission option by clicking the trash can icon on the right of the option.
6. Click **Apply** when done or **Cancel** to dismiss your changes.

In order to add/change generators for an individual structure, a user also has to have *Edit Generators* access (see page 362)for that structure.

Users who have *Jira Administrators* global permission are always allowed to change generators.

5.5.5 Changing Permissions to Configure and Run Effectors

There are three levels of permissions for Effectors(see page 154):

- **Permission to Configure Effectors** - by default, anyone with access to Automation(see page 480) can add and configure Effectors.
- **Permission to Execute Effectors** - by default, anyone with view access to a structure (except anonymous users) can run any Effectors within the structure.
- **Permission to Execute Effectors on Query Results** - this permission is required to run effectors that were inserted by a Structure Inserter(see page 128).

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

You can restrict users' ability to access/run Effectors based on user groups and/or project roles. To do so:

1. Navigate to **Administration | Structure | Configuration**.
2. Scroll to the appropriate Effector permission at the bottom of the page. Click the appropriate "Select who can..." button.
3. If you choose to assign permission based on **Users in selected groups/roles**, use the **Add Group/ Role** section to add one or more required user groups or project roles:
    a. Use the drop-down selectors to choose either the **Group** or **Project** option.
    b. Choose the required group name or project/role combination. (*Note:* If **project** is set to "Any", this means the user should be in the specified role for any of the structure-enabled projects(see page 477).)
    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.

5.6 Changing Structure Defaults

Jira administrators can adjust a number of Structure "defaults", settings that apply when the user does not specify a more specific request or option.

- Changing Default Structure(see page 482)
- Changing Default View Settings(see page 482)
- Changing Default Options for the Issue and Project Pages(see page 483)

5.6.1 Initial Configuration

When Structure is installed, the defaults are configured as follows:

System Default Structure	None
Project Default Structure	None
Default Views Menu	Preinstalled views: **Basic**, **Planning**, **Tracking**, **Triage**, **Entry** (on all pages)
Default View	**Basic View**
Auto-switch (Issue Page)	Structure with the displayed issue.
Auto-switch (Project Page)	**Off** - Show the last viewed structure.

Keep structure when navigating	**On** - When going from structure widget to an issue page, show the same structure.
Auto-minimize Structure Panel	**On** - Structure panel is initially minimized if the issue is not in the structure.

5.6.2 Changing Default Structure

The default structure(see page 357) is automatically selected when the user opens Structure Board(see page 83) for the first time. It is also shown when a user views the Structure widget from an issues page, if the Auto-switch option(see page 88) is set to *default structure*.

You can change the default structure for the entire Jira instance, as well as for specific projects.

5.6.2.1 Changing System-level Default Structure

The system-level default structure is used when no other structure has been selected or the Auto-switch option is set for issue pages (and a project-level default structure has not been specified).

1. Open the **Administration | Structure | Defaults** menu.
2. In the **System Default Structure** section, click **Change**.
3. Select the default structure and click **Apply**.

Make sure users have View permissions(see page 362) to the default structure. Users without view permissions will receive an error when the default structure should be displayed.

5.6.2.2 Changing Project-level Default Structure

You can select a default structure for specific projects.

1. Open the **Administration | Structure | Defaults** menu.
2. Locate the project in the **Project Default Structures** section and click **Change**. If you can't find the structure, un-check the **Show only projects with overridden default structure** checkbox.
3. Select a structure and click **Change**.
4. Or, select **Use system default** to remove the project-level default.

Project administrator can also change the project-level structure from the **Structure** tab on the project administration page, or from the options pop-up window on the **Structure** tab on the user's project page.

5.6.3 Changing Default View Settings

View settings determine which views are offered to the users in the Views Menu (on the Structure Board and other pages with Structure widget). The default view settings apply to all structures that don't have customized view settings(see page 360).

To change default view settings:

1. Open the **Administration | Structure | Defaults** menu.
2. In the **Default View Settings** section, click **Change**.

3. Modify the default settings - for details, see Customizing View Settings(see page 360).
4. Click Apply

5.6.4 Changing Default Options for the Issue and Project Pages

A number of options define how Structure behaves on issue pages(see page 88) and project pages(see page 90). When the user opens those pages for the first time, the default settings apply. These settings are adjustable by the Jira administrator.

To change the defaults:

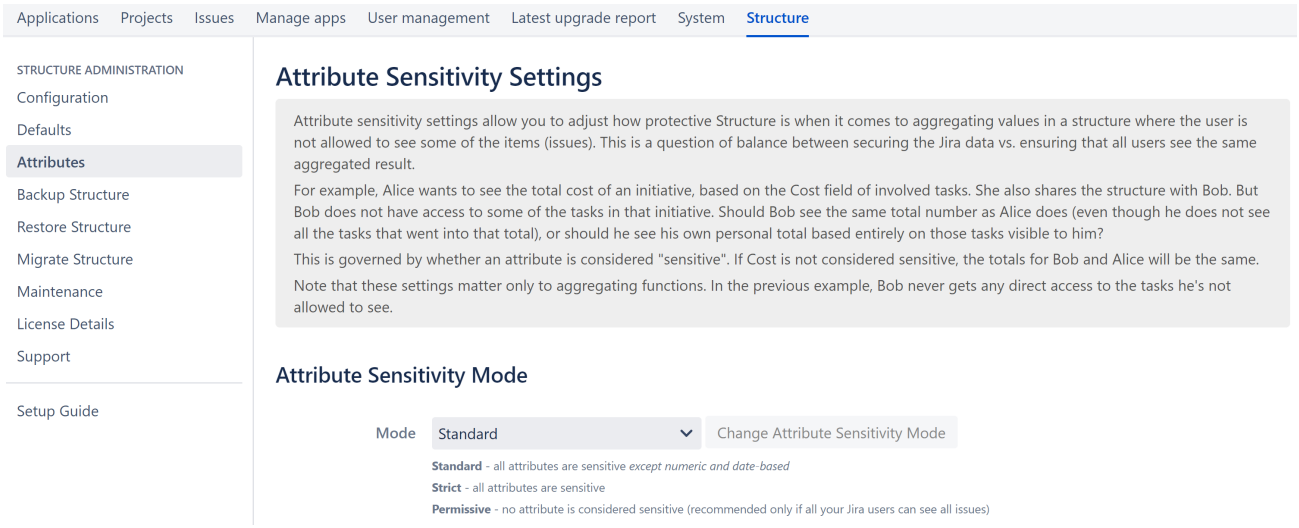
1. Open the **Administration | Structure | Defaults** menu.
2. Scroll down to **Structure User Interface Defaults** and click **Change**.
3. Make the changes and click **Change** again.

Option	Description	See Also
Auto-switch (Issue Page)	Lets you select which structure is initially displayed on issue pages.	Structure Options for the Issue Page(see page 88)
Auto-switch (Project Page)	Lets you select which structure is initially displayed on Project pages.	Structure on the Project Page(see page 90)
Keep Structure Selection When Navigating	If turned on, clicking an issue in the Structure Widget(see page 83) will open that issue's page with the same structure. If this option is turned off, the structure displayed will be based on the Auto-switch setting.	Structure Options for the Issue Page(see page 88)
Auto-minimize Structure Panel	If turned on, the Structure panel on the issue page will be initially minimized if the selected structure does not contain the displayed issue.	Structure Options for the Issue Page(see page 88)

If the user changes these options, those changes are preserved and are applied instead of the defaults for that specific user.

5.7 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.



5.7.1 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.

5.7.2 Setting the Attribute Sensitivity Mode

To set the Attribute Sensitivity mode for Structure, go to **Administration > Structure > Attributes**.

In the Mode field, select one of the following:

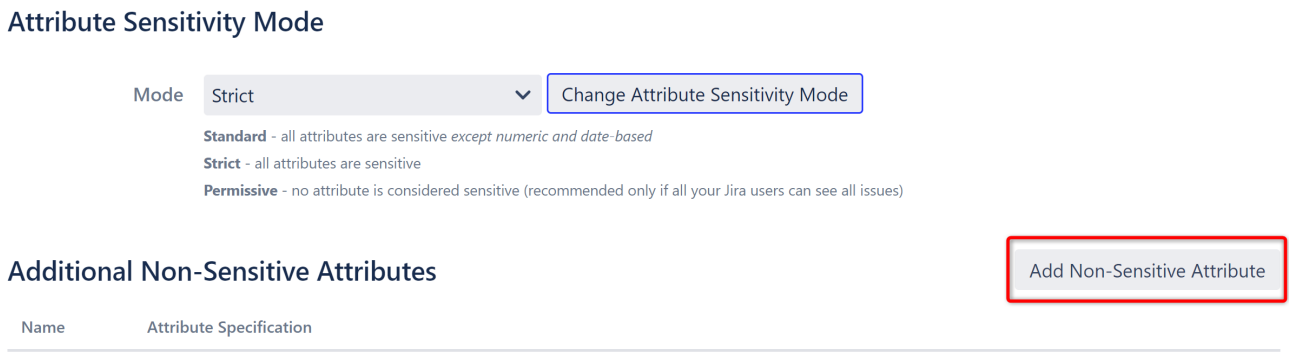
- Standard - sets all attribute fields **except numeric and date fields** to sensitive
- Strict - sets **all** attribute fields to sensitive
- Permissive - sets no attribute fields to sensitive

If an attribute is marked *Sensitive*, aggregate functions that include that attribute will only include values from issues the user has permission to view.

> ⚠ Permissive sensitivity does not apply to the Last Comment field. When aggregated in a formula, users will still only see comments for issues they have permission to view.

5.7.3 Assigning Specific Non-Sensitive Attributes

You can specify certain attributes to be non-sensitive, regardless of your sensitivity mode. To do so, go to the the Additional Non-Sensitive Attributes section and click the **Add Non-Sensitive Attribute** button.



Next select the attribute you wish to make non-sensitive. When you select from the **Attribute** dropdown, the **Format** and **Spec** details will be updated automatically. You can also enter these manually to customize the attribute requirements or enter an attribute not available in the dropdown.

Be careful! When you add non-sensitive attributes, you allow those values to be included in aggregate results for all issues, even if the user does not have access to an issue. The user will not **see** the specific value for issues they do not have access to, but it may be possible for them to deduce the value by comparing the aggregate value to values they can see.

5.8 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

5.8.1 Using Structure Backup

Structure can use a backup file in two ways:

- **Full structure restore.** This operation replaces all existing structure data (if any) with the data stored in the backup file. This operation refers to issues and other items by their numeric IDs (*not* issue keys!), so the issues must be present in Jira before this operation is run, and issue IDs must be the same as they were at the time the Structure backup file was created.

  Issue IDs are preserved if the Jira instance is fully restored from a backup with the **Restore System** command. Issue IDs are *not* preserved if the issues are moved to another Jira instance with Jira's **Project Import** feature – in this case, you should use the migration option below.

- **Migration / partial import.** This operation lets you restore one or more structures backed up from a different Jira instance (assuming the issues have been moved over with the Jira's **Project Import** command). It also allows you to merge the backup structure data with the structure data already existing on your Jira.

  A structure in a backup file cannot be restored if it refers to issues in a project that is not present in the Jira instance.

- Backing Up Structure(see page 486)
- Restoring Structure from Backup(see page 487)
- Migrating Structures(see page 488)

5.8.2 Backing Up Structure

Backing up Structure saves the existing structures, their configuration, hierarchies and other Structure data. Structure backup does not save the issues themselves or other Jira data - see Structure Backup, Restore and Migration(see page 485).

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.

5.8.3 Restoring Structure from Backup

Restoring Structure from a backup brings back the structures, automations, synchronizers, views and other data created at the moment of backup.

> ⚠ Restoring Structure will not affect issues in any way or restore them. The issues that make up the hierarchy should already exist in Jira. If you do full restore, then you need to run the standard Jira data restore first - see Structure Backup, Restore and Migration(see page 485).

> ⚠ The issues and other items in the structures are identified by their internal numeric ID. If you have transferred issues via Jira's Project Import, issue IDs have changed, and you will need to use Structure Migration(see page 488) 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.

5.8.3.1 Downgrading from Structure 5.0 or Later

The introduction of Manual Adjustments(see page 175) 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.

5.8.4 Migrating Structures

Migrating structure data lets you import one or more structures from a different Jira instances after you have imported projects with Jira's Project Import operation. In addition, you can add structures from a backup file to those that are already present in Jira.

> ⚠ Migrating structure will not affect issues in any way. The issues that make up the hierarchy should already exist in Jira. You may need to run Jira Project Import or the standard Jira data restore first - see Structure Backup, Restore and Migration(see page 485).

> ⚠ During migration, the issues in the structures are located in Jira by their issue keys. A structure cannot be migrated if it refers to issues from a project that is missing in Jira.

To migrate structures from a backup:

1. Navigate to **Administration | Structure | Migrate Structure**.
2. Enter the full path to the structure backup file (either *.xml* or *.zip*).
3. Click **Select Structures To Restore**.
4. Select the structures that should be restored. If there's an existing structure with same ID or name, select **Overwrite Existing** to replace the existing structure with the one from backup; otherwise the structure will be restored as a new structure, leaving the existing one unaltered.
5. Under the list of structures, you will see a list of additional restore options:

Restore Structure Permissions	If selected, Structure will attempt to restore the access permissions for the imported structures. This attempt may fail if the permission rules refer to users or groups not present in Jira. If you don't select this option, or if the attempt to restore permissions fails, the restored structure will have no permission rules. Jira administrators can configure permission rules through the Manage Structures(see page 354) page.
Restore Synchronizers	If selected, the synchronizers for selected structures are restored. ⚠ Synchronizers configuration is imported as-is, and might not make sense on a new Jira instance. After you have restored synchronizers, please visit the Synchronization Settings(see page 514) page to check if the synchronizers are configured correctly.

Restore Structure History	If selected, structures are imported along with their history (if it is present in the backup file). If not selected, structures will have no history.
Restore User Favorites	If selected, the plugin will try to restore "favorite" marks made by users for the selected structures.
Restore Views	If selected, all views from the backup files will be restored. If there's a conflict and a view with a given ID already exists, Structure will first verify if the view being restored is different from the one in the system. If it is, Structure will restore the backup view as a new view with a different ID.
Restore View Settings for Structures	If selected, view settings(see page 360) for the selected structures will be restored.
Restore Configuration	If selected, the global app permissions(see page 478) will be restored. This includes the list of projects, for which Structure is enabled, permissions to use or create structures, permissions to use Automation, etc.
Restore Dark Features	If selected, the dark features settings(see page 499) will be restored.

6. Click **Restore Selected Structures**.
7. You will see the Process Status page. Once the restore process is completed, the **Show Result** button will take you to the result page, where you'll be able to see the result and any warning messages.

After structures have been migrated, open the **Structure | Manage Structure** page to verify that your structures were restored successfully.

⚠ As of version 3.3, Migrate Structure does not support Structure.Testy or Structure.Pages data.

5.8.4.1 Owners and Permissions

When migrating structures, the following ownerships and permissions are included in the migration:

- Structure owners
- View owners
- Structure permissions
- "Apply Permissions" rules, if the structures they refer to are migrated
- When a permission rule cannot be migrated, Structure logs a warning and skips this specific rule

⚠ Backups created in a version of Structure prior to 6.2 do not contain usernames and project role names. When trying to migrate from an older backup file, ownerships and permissions that require this information will not be included.

5.9 Automatic Structure Maintenance

5.9.1 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.

5.9.2 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.

5.9.3 Maintenance Tasks

Backup Structure data	Creates a backup of the Structure database in the `export` sub-directory under Jira home. Parameters: • Include history – if checked, full structure change history will be included in the backup. If you have a lot of changes in structures, this setting may cause the backup to take some time, and the backup file to be large. If you don't need a history of structure changes, it is advised to turn this option off. ✓ It is recommended that you maintain separate Structure backups, even though Structure data is backed up with Jira's normal backup, because this will allow you to restore from that data without rolling back changes in Jira.
Delete old backups	A backup is considered old if it is not among *X* latest backups (*X* is specified by the first parameter of this task) **and** it was made earlier than *Y* days ago (*Y* is specified by the second parameter). This task removes all such backups made by the Backup task. Parameters: • Always keep *X* latest backups • Always keep backups made during the last *Y* days
Optimize favorites	If a user marks a structure as their favorite(see page 357), Structure will keep this mark, even if the user is later deleted from Jira, and the popularity number of the structure(see page 357) will still accounts for that user. This task removes marks made by users no longer in Jira and recounts the structure popularity.
Optimize structures	If an issue is added to a structure and then deleted from Jira, the structure will still contain a reference to this issue (although it will not display it). This task removes references to deleted issues and other items that have become permanently unavailable.
Optimize view settings	If a view(see page 341) is deleted, some structure view settings(see page 360) may still reference it, and a blank view named **? (Unknown View)** will be shown in its place. This task removes references to deleted views.
Optimize synchronizers	This removes an data related to synchronizers of a deleted structure.

Delete old synchronizer audit log records	This removes old records from the Synchronizer Audit Log(see page 490), clearing up space in the database. Parameters: • Keep records for the last *X* days. If you set X to 0, the maintenance procedure will remove all records.
Reindex change history	Currently does nothing. This task has remained as an option since Structure 2. Its purpose may be restored in a later version of Structure.
Optimize structure perspectives	Removes old perspectives(see page 92) that haven't been used by anyone for a certain amount of time. Parameters: • Delete perspectives that were not used during the last *X* days
Reindex structures	Clears and recalculates issue-to-structure index, used to define which structures contain a specific issue. (Issues added with Automation(see page 122) are not counted.)
Delete old change history	The task removes old records from the change history. A history record is considered old if the change was made earlier than X days ago (X is specified by the first parameter) **and** it is not among the Y latest history records for the structure where the change was made (*Y* is specified by the second parameter). Parameters: • Always keep change history for the last X days • Always keep the Y latest changes per structure

5.9.4 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.

5.10 Workflow Integration

5.10.1 Structure Workflow Validator

Structure Plugin adds a new workflow transition validator[86] to JIRA. This validator blocks the transition if the issue doesn't match an S-JQL query(see page 266). 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.

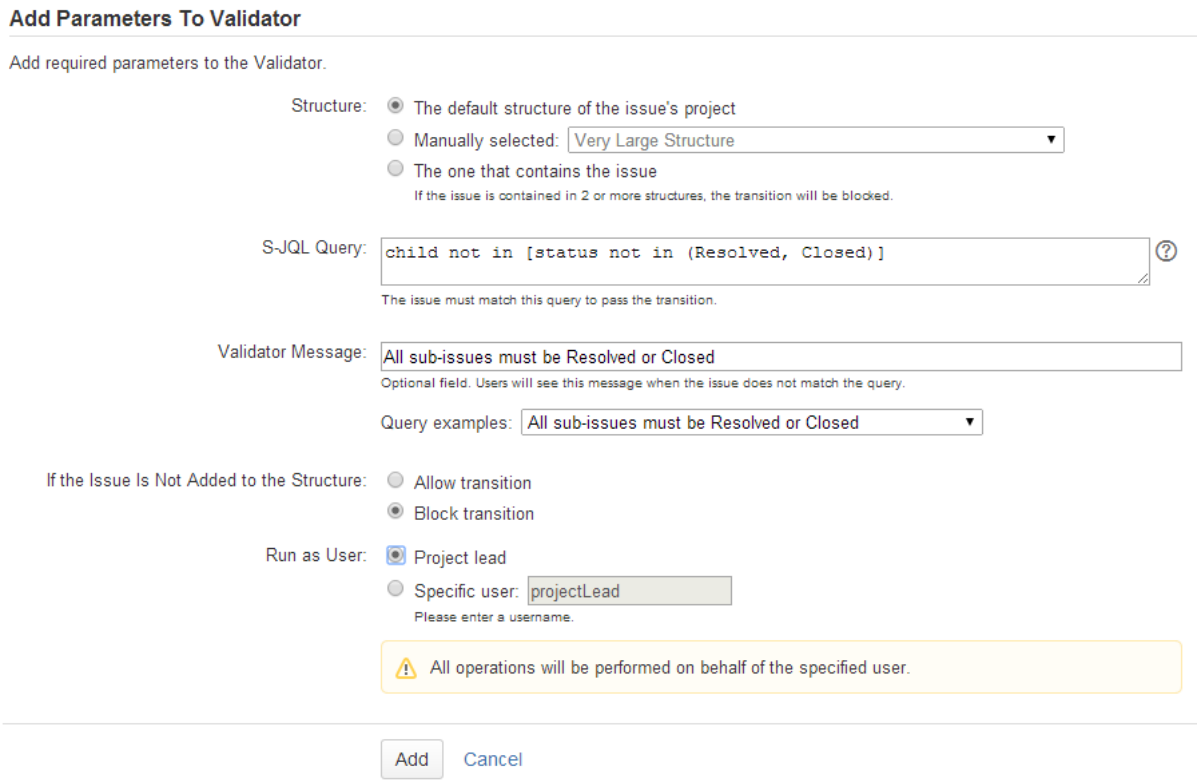
To add the Structure validator to a workflow:

1. Create a draft of the workflow and open the **Add Validator to Transition** dialog. (For more information, please refer to the Jira documentation[87].)

86 https://confluence.atlassian.com/display/JIRA/Advanced+workflow+configuration#Advancedworkflowconfiguration-Whatareconditions,validators,andpostfunctions?

87 https://confluence.atlassian.com/display/JIRA/Advanced+workflow+configuration#Advancedworkflowconfiguration-validatorAddingavalidator

2. Select the **Issue Matches Structure Query** validator. A configuration window will open:



3. In the **Structure** field, specify how the validator should select the structure to check. It can either be a manually-selected structure, or it may depend on the issue being checked (the default structure of the issue's project or the structure that contains the issue).

> ⚠ In Structure 3, the option to pick a structure that contains the issue being validated is no longer available.

4. In the **S-JQL Query** field, enter the S-JQL query that the issue should match in order to pass the transition.

> ✓ You can use one of the examples provided with the form. Just select an example in the **Query Examples** selector, and the corresponding query will be copied into the **S-JQL Query** field.

5. In the **Validator Message** field, enter an explanation message that users will see if their transitions are blocked by the validator.
6. In the **If the Issue Is Not Added to the Structure** field, select whether the transition should be blocked or allowed if the issue is not contained in the checked structure. (Or if the issue does not belong to any structure, in case automatic structure selection is chosen.)
7. In the **Run as User** field, select on behalf of which user the validator should run. It can either be a manually-selected user, or it may be the lead of the project of the issue that is being checked.

> ⚠ Running on behalf of a user means that the validator will only see issues and structures that are accessible to the specified user. The result of the validator check will depend on the permissions of the specified user and will not depend on permissions of the user who performs the transition.

5.10.2 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[88].

> 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.

5.11 Running Structure on Jira Data Center

Structure is a Data Center approved app(see page 561).

The following articles provide additional information, specific to the Data Center editions.

- Archived Projects and Structure(see page 496)

To learn more about the Data Center approved app program, see our Data Center Approved Apps FAQ(see page 561).

5.11.1 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.

5.11.1.1 Archived Issues in Structure

> Archived issues will not be visible in Structure – even if they were added manually, and even in the archived structures.

When you archive a project, check for structures that have issues from that project. Contact the users to see if they are okay with the archived issues disappearing from the structures.

88 https://confluence.atlassian.com/display/UPM/Viewing+installed+add-ons#Viewinginstalledadd-ons-AboutAdd-onModules

5.11.1.2 Restoring an Archived Project

When you restore a project from archive, you run a project reindex. After the reindex has finished, the issues should reappear in Structure.

Since Structure has internal caching for which issues are visible to which users, it might take a few minutes after the reindex completes for issues to reappear in Structure. To force the internal caches to clear and see the restored issues in structures immediately, use the "force reload" action from your browser (Shift+Reload).

5.12 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[89]

5.12.1 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**

5.12.2 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[90] and our privacy policy[91].

89 https://wiki.almworks.com/download/attachments/32222999/statisitics.txt?api=v2&modificationDate=1491358938000&version=1
90 http://almworks.com/EULA-Structure.pdf
91 http://almworks.com/company/legal/privacy.html

5.12.3 Structure Usage Statistics

Recent statistics can be viewed at any time.

1. Go to **Administration** | **Structure** | **Support** and scroll to the **Anonymous Usage Statistics** section near the bottom of the page.
2. Click **View Current Statistics** to see the Structure usage data recorded for the past 30 days.

> Usage statistics are always recorded, but are only sent to to us if the '**Send anonymous usage statistics**' option is enabled. Sharing this anonymous data with us helps us improve our product, but if you do not wish to, simply uncheck this option.

> Jira administrator rights are required to access these statistics.

The log shows statistics related to different actions in structures - number of created items, description of existing generators, number of clicks made on specific buttons, etc.

Among this data, there is:

- The *activeUser* parameter, which shows the number of users who performed meaningful actions in any structure - created new items, added/removed/moved items, etc. Users who only viewed structures, without changing anything, are not counted.
- The *structureCount* parameter, which shows the total number of structures in the system.

5.13 Structure Files Location

5.13.1 Structure Files

5.13.1.1 `$JIRA_HOME/structure`

Structure keeps most of its data in the `structure` sub-directory under the JIRA home directory[92].

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.

5.14 Turning Off Optional Features

Some features in Structure are designed as modules and can be safely turned off. This can be done to remove unnecessary functionality or limit the exposure to Structure for specific users.

92 http://confluence.atlassian.com/display/JIRA/JIRA+Home+Directory

If your aim is to limit the exposure to Structure, consider restricting permissions to specific groups of users - see Structure Best Practices(see page 459).

While it is easy to disable many Structure modules, we don't recommend touching any modules except those listed in this article to ensure stability of Structure and your Jira application.

To turn off a module:

1. Open the Add-on Manager by navigating to **Administration | Add-ons | Manage Add-ons**.
2. Locate the Structure add-on and expand its row.
3. Click the link that looks like the following: "309 of 310 modules enabled." (Numbers may vary.)
4. Use the Search feature of your browser to find the module by its name (provided below.)
5. Click the **Disable** button to the right of the module name.

You can always turn the feature on later by clicking the **Enable** button.

Feature	Module name	Effect of disabling this module
Activity Streams	`Structure (structure-activity-provider)`	The Activity Streams provider and Structure-related updates are removed from the following places: • Activity Stream gadgets(see page 381), • Activity tab on the issue page • Activity tab on the user page • Activity tab on the project page
Structure on the Issue Page(see page 85)	`web-resource:Issue Page Decorator (adjustIssue)`	The Structure section is removed from the issue page.
Structure on Agile Boards(see page 91)	`web-panel:GreenHopper tab (greenhopper-tab)`	The Structure tab is removed from the issue details panel on the Agile board.

5.15 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.

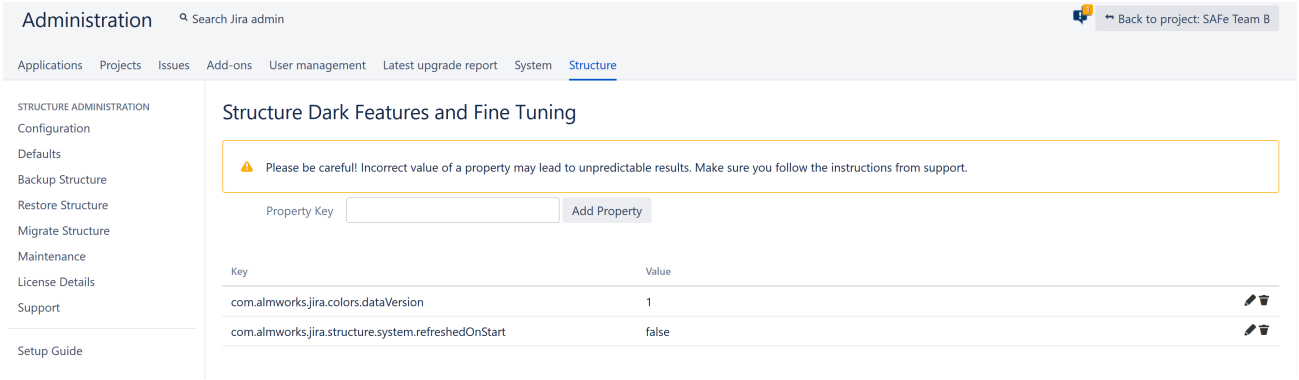
- Setting Application Properties with the Structure Dark Features and Fine Tuning Interface(see page 500)
  - Guidelines for Adding/Removing Property and Values(see page 500)
- Setting System Properties(see page 501)
  - Setting System Properties on Startup(see page 501)
  - Setting System Properties with Script Runner(see page 501)
- Available Dark Features(see page 502)
  - Structure size limit(see page 502)
  - Structure Automation limits(see page 502)

- Automation Defaults(see page 502)
- Manual adjustments(see page 503)
- Hidden Issue Links(see page 503)
- Index Consistency Checks(see page 503)
- Alternative initial values for project/type when creating an issue in dialog(see page 503)
- Manage Permissions for Groups Without Belonging to Them(see page 504)
- Resolved icon(green tick)(see page 504)
- Structure Export Timeout(see page 504)
- Time in Status - Refresh Period(see page 505)
- Time Tracking on the Issue Page and Issue Details Page(see page 505)
- Link-related changes made by Generators or Synchronizers(see page 505)
- Synchronizers(see page 506)

5.15.1 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.



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

5.15.1.1 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, it's property value is set to the default value:
  - If the property was added with our admin interface, the value is set to empty value and the property is removed from the table after a page refresh.
  - If you set the value to empty (without deleting the property), the property will not be removed.

5.15.2 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.

5.15.2.1 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[93]. You will need to restart Jira for the properties to take effect.

5.15.2.2 Setting System Properties with Script Runner

You can also set system properties using the Script Runner[94] 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.

93 https://confluence.atlassian.com/display/JIRA/Setting+Properties+and+Options+on+Startup
94 https://marketplace.atlassian.com/plugins/com.onresolve.jira.groovy.groovyrunner

5.15.3 Available Dark Features

5.15.3.1 Structure size limit

Property	Default	Explanation
`com.almworks.jira.structure.AOBasedStructureManager.forestSizeLimit`	`1000000`	The maximum number of rows that one structure can contain.

5.15.3.2 Structure Automation limits

Property	Default	Explanation
`structure.gfs.generationTimeHardLimit`	`600`	The maximum amount of time that can be spent for Structure generation (in seconds).

5.15.3.3 Automation Defaults

Property	Default	Explanation
`structure.generator.defaults.disableUpdates`	`false`	When adding generators: • If set to "false" (default) - the "allow changes" box is initially checked. • If set to "true" - the "allow changes" box is initially unchecked.

5.15.3.4 Manual adjustments

Property	Default	Explanation
`structure.gfs.manualAdjustments.enable`	`true`	Setting this property to false will disable manual adjustments for the entire Jira Instance. All adjustment-related UI elements and controls will disappear. Existing manual adjustments will be kept in the database, but will not be applied.
`structure.gfs.manualAdjustments.maxAdjustmentsPerStructure`	`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.
`structure.gfs.manualAdjustments.maxAdjustmentsPerAction`	`200`	The maximum number of manual adjustments per one user action. If this limit is exceeded, the action will be aborted without making any changes.

5.15.3.5 Hidden Issue Links

Property	Default	Explanation
`structure.feature.hiddenLinks.enabled`	`false`	Set to true to enable support for hidden issue links.

5.15.3.6 Index Consistency Checks

Property	Default	Explanation
`structure.indexConsistencyChecker.disabled`	`false`	Set to true to disable periodical checks of Lucene index consistency.

5.15.3.7 Alternative initial values for project/type when creating an issue in dialog

Normally, when the user creates new issues through dialog, Structure remembers the selected project and issue type and offers those the next time by default. This dark feature enables a different algorithm, which used to work

in a previous version of Structure: the initial project and issue type are taken from the issue that was focused when "+Create" or "+Next Issue" was pressed.

System property	`structure.feature.altInitialValuesInDialog`
Options to add in setenv.sh / setenv.bat	`-Dstructure.feature.altInitialValuesInDialog=true`
Internal feature name	`altInitialValuesInDialog`
Introduced in version	2.11.0

5.15.3.8 Manage Permissions for Groups Without Belonging to Them

By default, users can only manage permissions for groups they are a part of.

Property	Default	Explanation
`structure.permissions.allowAllUserGroups`	`false`	Setting this property to "true" allows users to manage any group, whether they belong to that group or not.

5.15.3.9 Resolved icon(green tick)

Property	Default	Explanation
`structure.doneAttribute.byResolution`	`false`	false - the "Resolved icon" shown when the Resolution field of an issue is non-empty true - the "Resolved icon" is shown when the issue is in a Done status category (StatusCategory.COMPLETE)

5.15.3.10 Structure Export Timeout

For large, complex structures, it may take a long time to capture all the details of a structure required for exporting. In order to reduce the amount of time and resources required for export, after a set time Structure will attempt to use a cached version of the structure for the export. If this is not possible, it will continue trying to compile live data for a set time before the export fails.

Property	Default	Explanation
`structure.export.trySnapshotTimeout`	`120`	In seconds, the amount of time Structure will wait before attempting to export a cached version of the structure. Note: when a cached version is exported, a warning is added to the export.
structure.export.forestLockTimeout	`600`	In seconds, the total time allotted before the export fails.

⚠ Minimum time is 60 or 90 seconds, and it can take up to 60 seconds longer than the configured timeout, because lock time is measured in 30s increments (by default), and there are 2 or 3 loops where the exporter waits, depending on the attributes and forest spec.

5.15.3.11 Time in Status - Refresh Period

Property	Default	Explanation
`structure.timeinstatus.refreshPeriod`	`3600000`	Sets update period for Time in Status columns. Value is in milliseconds. Time is Status is updated any time a status change occurs in Jira or once per hour if status remains unchanged. This option allows you to update the Time in Status more or less often, when issues remain in the same status.

5.15.3.12 Time Tracking on the Issue Page and Issue Details Page

When viewing an Issue page in Jira or the Issue Details page in Structure, the Time Tracking section provides up-to-date time tracking details, and allows you to include time tracking details for sub-issues within the calculations. If these pages are taking too long to load, disabling this feature may help.

Property	Default	Explanation
`structure.issuePage.showTimeTracking`	`true`	false - disables the Structure Time Tracking section on the Issue page and Issue Details page.

5.15.3.13 Link-related changes made by Generators or Synchronizers

By default, when a generator (see Automation(see page 122)) or Synchronizer creates, deletes or changes a link, the change appears immediately on an Issue Page, but it is not immediately indexed in Jira. Additionally, the change is not written to history, so it is not possible to review the changes on an issue's History tab.

This setup prevents link changes from negatively impacting system performance (a particular risk when bulk changes are made to issue links); however, there may be times when these types of changes need to be indexed immediately and/or appear in an issue's history. To enable this, simply update the following properties.

Property	Default	Explanation
`structure.bulkLinkProcessor.useLinkManager`	`false`	true - enables the immediate re-indexing after every change, writes changes in the history and creates a Structure-side event for each link update (enabling other apps to see that Structure made the change).
structure.sync.links.force.reindex	`false`	true - enables immediate indexing.

5.15.3.14 Synchronizers

Synchronization(see page 514) lets you keep Structure issue hierarchy in sync with some other issue properties. This dark feature is disabled by default.

Property	Default	Explanation
`structure.feature.synchronizers.enabled`	`false`	Set to true to enable Synchronizers within Structure.

Synchronizer Cycle Guard

The cycle guard(see page 522) is a component that detects conflicting synchronizers and prevents them from cycling forever, overriding each other's changes. The table below describes the system properties that control the cycle guard.

Property	Default	Explanation
`structure.sync.guard.disable`	`false`	Set to `true` to disable the cycle guard. Conflicting synchronizers will not be prevented from running forever. **Not recommended.**
`structure.sync.guard.maxAutosyncsWithoutUserChanges`	`10`	The maximum number of times that a synchronizer is allowed to run, processing the changes generated by another synchronizer. If this limit is exceeded, the two synchronizers are considered to be in conflict.

Property	Default	Explanation
`structure.sync.guard.stop.disable`	`false`	If `true`, conflicting synchronizers will not be disabled automatically. The cycling may repeat after a user-generated change.
`structure.sync.guard.email.owner.disable`	`false`	If `true`, the cycle guard will never send e-mail notifications to synchronizer owners.
`structure.sync.guard.email.admin.disable`	`false`	If `true`, the cycle guard will never send e-mail notifications to Jira administrators.
`structure.sync.guard.email.admin.cycles`	`10`	The minimum number of times a cycle must be detected for a synchronizer before an e-mail notification about that synchronizer is sent to Jira administrators. The counter is reset when a synchronizer is automatically disabled, so if this number is greater than 1 and automatic disabling is on, the administrators will not be notified.

5.15.4 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(see page 499). 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[95].

5.15.4.1 Learn more about Structured Predictions

- Configuring Structured Predictions(see page 508)
- Using Structured Predictions(see page 510)
- Structured Predictions FAQ(see page 513)

5.15.4.2 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

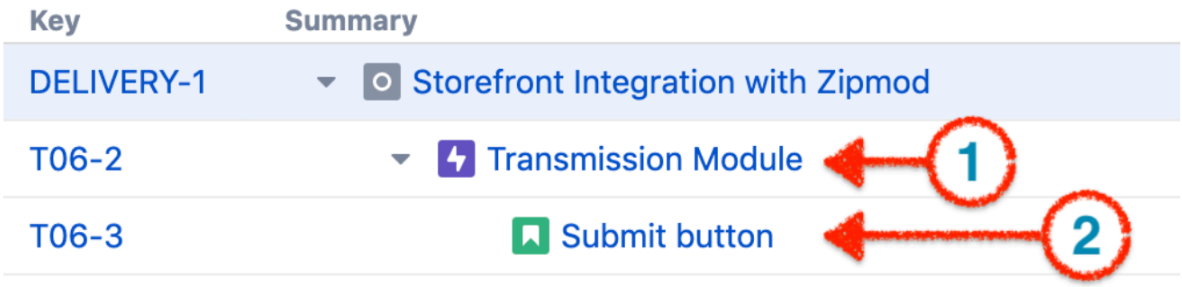
Structured Predictions uses Structure for Jira[96] from ALM Works to define data sources used for predictions. Configuration options allow for some flexibility in how the Structure data sources are configured, but some guidelines will make it easier to align your data with the Structured Predictions 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.

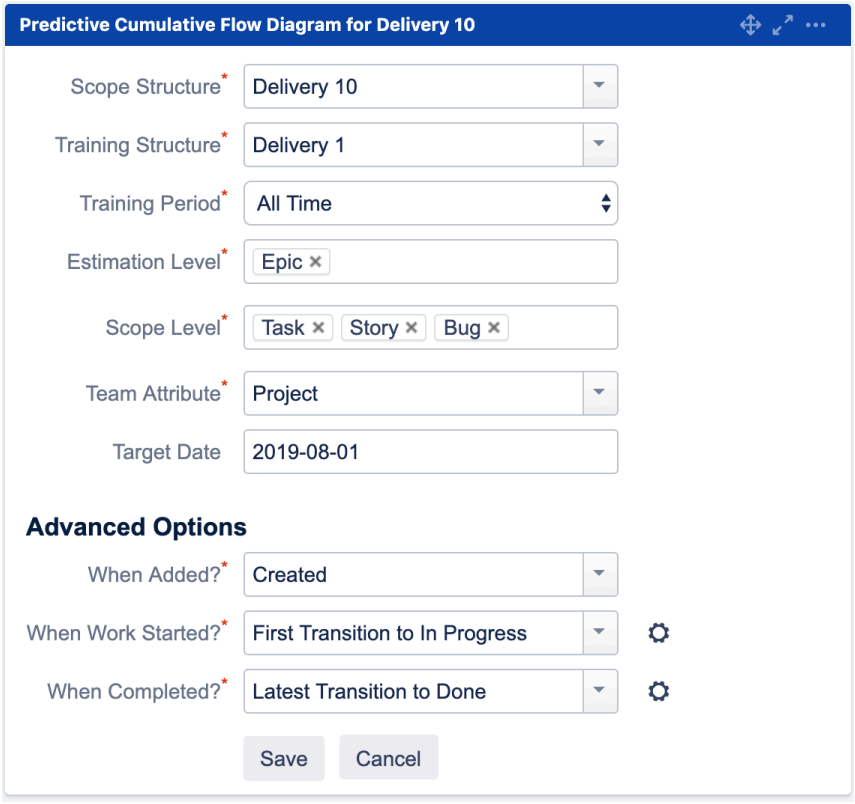
A structure that is configured to show Epics as estimates and Stories under those epics to show refined scope is in good shape to feed directly into Structured Predictions. A structure configured this way will look something like the figure below.

95 mailto:solutions@almworks.com
96 http://alm.works/structure



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.



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, than 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(see page 499). 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[97].

5.15.4.3 Using Structured Predictions

Reading the Structured Prediction Gadget

Structured Predictions generates a Predictive Cumulative Flow Diagram and provides specific metrics for each team.

97 mailto:solutions@almworks.com

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(see page 508)) 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(see page 508). 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 throughout. This is the standard deviation for the underlying data used to determine the throughout. 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(see page 499). 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[98].

5.15.4.4 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(see page 508).

How do I get Structured Predictions?

Structured Predictions is what we refer to as a Dark Feature(see page 499). 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[99].

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[100] 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[101] 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[102] to set up a meeting to review your specific situation.

98 mailto:solutions@almworks.com
99 mailto:solutions@almworks.com
100 mailto:solutions@almworks.com
101 mailto:solutions@almworks.com
102 mailto:solutions@almworks.com

5.15.5 Synchronization

Dark Feature - Synchronizers are hidden by default

If there are no synchronizers installed on your system, this feature will now be hidden from users. To enable synchronizers, set **structure.feature.synchronizers.enabled** system property to **true**. See Advanced Configuration and Dark Features(see page 499) for more information.

If you currently have synchronizers installed, the feature will remain visible.

Synchronization lets you keep a structure issue hierarchy in sync with some other issue properties. For example, you can enforce the rule that Jira sub-tasks should always be placed under their parent in the structure, or that there should be an issue link from parent issue to each sub-issue.

Synchronization can also be run once to perform a one-time update of the structure (Import) or one-time update of the issues based on the structure (Export).

Synchronization is an extendable system that allows Jira add-ons to provide their own synchronizers. The following synchronizers are supplied with Structure:

- Sub-Tasks Synchronizer(see page 524) places Jira sub-tasks under their parent issues in the structure.
- Links Synchronizer(see page 528) makes sure sub-issues are linked to their parent issues with a specific link type. It can also be used to reconstruct structure from links.
- Filter Synchronizer(see page 524) populates structure with issues that pass a saved filter. It can also be used to remove issues from a structure when they no longer pass the filter.
- Agile Synchronizer(see page 531) works to sync the Agile ranking of issues with their position in the structure and to make it easier to assign stories to epics using Structure.
- Status Rollup Synchronizer(see page 534) propagates issue status upwards. For example, it can make parent issue Resolved if all sub-issue are resolved.

One-time synchronization works when you run Export(see page 516) or Import(see page 515), or when you run a Resync(see page 520). 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.

5.15.5.1 Import Synchronization

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

When you run an **Import** synchronization, you get a set of issues which should be in the structure and/or information on how they should be arranged in a hierarchical list. This information is then applied to an existing structure.

For example, you can use a Filter Synchronizer(see page 524) to add All Open Issues to a structure (or the results of whatever Saved Filter you have), or you could use the Agile Synchronizer(see page 531) to rearrange issues in the structure according to their rank and epic.

> To run Import, you must have **Control** permissions on the structure and additional permissions may be required by a specific synchronizer.

To import hierarchy into a structure:

1. Open the Manage Structures(see page 354) page using the top navigation Structure menu.
2. Select the structure you'd like to import into and click the **Import** link.

   > If you don't see the **Import** link in the Operations column, you may not have Control permissions for this structure.

3. Select a synchronizer from the drop-down list and configure the import parameters.

   > If there are no synchronizers in the drop-down list, either none are currently installed or none of the installed synchronizers support import into a structure.

4. Enter the synchronizer parameters. Each synchronizer has its own parameters, so please refer to the specific synchronizer documentation(see page 523).
5. Click **Run Import**. When you start the import, the synchronizer will analyze the existing data and possibly update the entire structure.
6. After you click Run Import and confirm the operation, a job status page is shown. When the job is marked as Finished, the synchronization is done, and you can view the results by opening the affected structure.

> When import is run, it runs under your user name and with your permissions. If you don't have enough permissions to read values somewhere else or to view issues you'd like to import, you may not see the expected result.

Import(see page 515) and Export(see page 515) are actually a one-time Resync(see page 520). 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!

5.15.5.2 Export Synchronization

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

When you run an **Export** synchronization, you use the hierarchy from a structure to create, update or delete issue attributes or features, based on the hierarchy that a specific synchronizer provides.

For example, you can use the Links Synchronizer(see page 528) to create issue links between sub-issues and their parents.

To run Export, you must have **Control** permissions for the Structure, and you will likely need some additional permissions, depending on which synchronizer you're going to use. For example, you have to have *Link Issues* permission when working with the Links synchronizer.

To export hierarchy from a structure:

1. Open the Manage Structures(see page 354) page using the top navigation Structure menu.
2. Select the structure you'd like to export from and click the **Export** link.

   If you don't see the **Export** link in the **Operations** column, you may not have **Control** permissions on this structure.
3. Select a synchronizer from the drop-down list and configure the export parameters.

   If there are no synchronizers in the drop-down list, either none are currently installed or none of the installed synchronizers support exporting from a structure.
4. Enter the synchronizer parameters. Each synchronizer has its own parameters, so please refer to the specific synchronizer documentation(see page 523).
5. Click **Run Export**. When you start the export, the synchronizer will read the current structure and apply it to whatever it syncs with.
6. After you have clicked Run Export and confirmed the operation, a job status page will be present. When the job is marked *Finished*, the synchronization is done and you can inspect the results.

When export is run, it runs under your user name and with your permissions. If you don't have enough permissions to make a certain change in Jira, the synchronizer will skip that change (a warning will be printed out in the server logs).

Import(see page 516) and Export(see page 516) are actually a one-time Resync(see page 520). 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!

5.15.5.3 Installing a Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

When you install a synchronizer on a structure, you make this structure automatically sync with something else (as opposed to doing a one-time import(see page 515) or export(see page 516)).

Here are a couple of examples of how that might work:

- With a Links Synchronizer(see page 528) installed and enabled, any changes someone makes to the structure will cause issue links to be created or deleted in Jira to match those changes.
- With a Filter Synchronizer(see page 524) in *Add* mode installed and enabled, when someone creates or changes an issue in Jira, if that issue now passes the selected saved filter, it will be automatically added to the structure.

When you install a synchronizer, you define its parameters. These parameters can be edited(see page 518) later.

To install a new synchronizer:

1. Open the **Manage Structure** page using the top navigation Structure menu.
2. Find the structure you'd like to sync. The **Sync With** column shows currently-installed synchronizers. Click the **Settings** link in that column.

   If you don't see the **Settings** link in the **Sync With** column, you may not have **Control** permissions for this structure.
3. The synchronization settings page shows detailed information about each installed synchronizer and lets you work with them. To proceed with the installation of a new synchronizer, select the type of synchronization and click **Configure and Install Synchronizer**.
4. Enter the synchronization parameters. Each synchronizer has its own parameters, please refer to the specific synchronizer documentation(see page 523).
5. Press the **Create** button to install the synchronizer. Though installed, the synchronizer is not enabled yet.
   a. Before synchronization is enabled, we recommend running Resync(see page 520) to sync the current state of the structure and Jira. To do so, press the **Resync and Enable** button after the synchronization is installed, or later use the same link on the synchronization settings page.
   b. If you need to enable synchronization without resyncing first, press **Enable without Resyncing**.
   c. You can enable and resync the synchronizer later from the synchronization settings page. Press **Done** if you don't need to enable the newly installed synchronizer now.

After a synchronizer is installed, it's not working yet - it must be **Enabled** to start monitoring the changes.

Import(see page 517) and Export(see page 517) are actually a one-time Resync(see page 520). 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!

5.15.5.4 Modifying Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

You can change the parameters of a synchronizer to alter how it works.

If the synchronizer is enabled (and working in the background), it needs to be turned off(see page 519) before you can make any changes, because changing a synchronizer's parameters may completely change the result of synchronization.

To edit a synchronizer:

1. Open the **Manage Structure** page using the top navigation Structure menu.
2. Find the structure with the synchronizer you'd like to edit. Click on the **Settings** link in the **Sync With** column.

   If you don't see the **Settings** link in the **Sync With** column, you may not have **Control** permissions on this structure.

3. Find the synchronizer you'd like to edit. Click the **Edit** link or the **Disable and Edit** link in the **Operations** column. The synchronizer will be automatically disabled.

   If you don't see the **Edit** link or the **Disable and Edit** link in the **Operations** column, the synchronizer is probably provided by a third-party plugin and does not support editing.

4. Set the new synchronizer parameters. Also, if you are a **Jira Administrator** and not the synchronizer owner, choose if you want to become the new synchronizer owner.
5. Press the **Apply** button. This will update the synchronizer parameters and make you the new owner of the synchronizer (unless you chose not to do so in the previous step). However, the synchronizer is not enabled yet:
   a. Before synchronization is enabled, we recommend running Resync(see page 520) 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.

5.15.5.5 Removing Synchronizer

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

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.

5.15.5.6 Turning Synchronizer On and Off

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

A synchronizer can be in one of the following states:

- **Disabled** - the background incremental synchronization is not running. You can run resync(see page 520) to do a one-time full sync.
- **Enabled** - the background incremental synchronization is running. When a change is detected, synchronization applies the change to the other part of the synchronous link as soon as possible, typically within several seconds.
- **Not Available** - the synchronizer is installed but it cannot run any synchronization. The possible reasons are changes in Jira configuration or lack of permissions from the user.

> Synchronizers are disabled by default. After installation(see page 517), they must be enabled, typically with a resync(see page 520).

Turning a Synchronizer On

To **enable** an inactive synchronizer:

1. Open the synchronization settings page for the structure.

2. Find the synchronizer and click the **Enable** link.
3. Alternatively, you can click **Resync and Enable** to resync(see page 520) 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.

5.15.5.7 Running Resync

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

A resync, or full resynchronization, is a one-time process activated manually by the user to sync current state of the structure and Jira.

> Resync scans all issues that may be affected by the synchronizer - contrary to the incremental synchronization, which inspects only issues that have been changed.

Resync Direction

When running a Resync, you need to choose it's direction.

Resync can be run:

- *from Structure* - Issues in Jira will be updated based on the structure hierarchy. This is what happens when you export from a structure(see page 516).
- *into Structure* - The structure hierarchy is updated (issues will be moved, added or removed), based on the current state of the issues in Jira. This is what happens when you import into a structure(see page 515).

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.

5.15.5.8 Synchronization and Permissions

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

When sync runs, the updates will be made on behalf of the user who installed the synchronizer! Any change that a synchronizer makes when reconciling data between Structure and Jira is performed on behalf of the user who created the synchronizer, not the user who ran the synchronizer.

This is really important to understand. Consider the following settings:

- You create a Structure and set up structure permissions(see page 362) so that anyone can edit the structure.
- You have Link Issues permissions on a project, and you install a Links synchronizer to have child issues linked to their parent issue.

Now anyone can edit the structure - add issues, remove issues and rearrange the issues in the structure. **Every change to the structure will lead to adding and removing links between the affected issues on your behalf - even if the user who changes the structure does not have Link Issues permission!**

For all existing synchronizers, the creator's username can be found in the Run as User column on the synchronization settings page. Before executing the transaction, Jira validates the user's permissions and then records the result together with the username in the log.

Changing Permission to Manage Synchronizers

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

By default, any logged-in user with Control access level(see page 362) for a structure can manage that structure's Synchronizers(see page 514). However, you can restrict this ability to one or more user groups.

To select who can manage synchronizers:

1. Navigate to **Administration | Structure | Configuration**.
2. Click **Select Who Can Control Synchronizers**.
3. Select whether synchronizers can be managed by **Anyone with control access to the structure** or by **Users in selected groups/roles**.
4. In the latter case, change the **selected groups/roles** list by selecting the second radio button and use the **Add Group/Role** section to add one or more required user groups or project roles. To set up the required property, use the drop-down selectors to choose either the **Group** or **Project** option, then choose the required group name or project/role combination and press the **Add** button to add it to the list. If **project** is set to "Any", this means that the user should be in the specified role for any of the structure-enabled projects(see page 477).
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.

5.15.5.9 Protection from Synchronizer Cycles

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

It is possible to accidentally create a pair of synchronizers that would contradict each other. For example, a sub-tasks synchronizer(see page 524) can be configured to put a sub-task under an issue, while a links synchronizer(see page 528) with the "links primacy" option would have to move it to the top level of the structure.

If both such synchronizers are enabled (i.e. perform automatic synchronization), they will end up in an endless cycle, processing and overriding one another's changes, forever. These situations are undesirable, because they put unnecessary load on the server and quickly fill up issue and structure histories with meaningless change records.

Structure is designed to detect and stop such infinite cycles. In the background, Structure keeps track of how many times each of the enabled synchronizers has been invoked to process the changes generated by another synchronizer. If this number passes a certain threshold (10 by default), and there were no user-generated changes between the invocations, Structure will flag this as a probable conflict, and perform one or more of the following actions, depending on the configuration:

- Prevent one of the synchronizers from running this particular time, but keep both synchronizers enabled.
- Disable both of the synchronizers involved in the cycle.
- Send e-mail notifications to the user (or users) who created the conflicting synchronizers.
- If the synchronizers are not automatically disabled and keep cycling, send e-mail notification to the Jira administrators (all users having the "System Administrator" global permission).

The default behavior is to disable the conflicting synchronizers and send e-mail notifications to the users who installed them.

> **How do I respond to a cycle warning?**
>
> If you've installed a Structure synchronizer and then receive a cycle warning e-mail from Structure, please take appropriate measures to reconcile the synchronizers – disable or reconfigure one or both of them. If the second synchronizer was created by a different user, you may need to cooperate with them to solve the problem. If you're not sure what to do, contact your Jira administrators or the ALM Works support team.

Jira administrators can configure the cycle guard as described in the Advanced Configurations Guide(see page 0).

5.15.5.10 Bundled Synchronizers

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

There are several bundled synchronizers included with Structure. Other synchronizers can be provided by other Jira add-ons.

- Sub-Tasks Synchronizer(see page 524) — The Sub-Tasks Synchronizer automatically places sub-tasks under their respective parent issues in the structure.
- Filter Synchronizer(see page 524) — The Filter Synchronizer automatically adds issues to a structure or removes issues from a structure based on a Saved Filter or a JQL query.
- Links Synchronizer(see page 528) — The Links Synchronizer maintains issue links between parent issues and child issues.
- Agile Synchronizer(see page 531) — 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.
- Status Rollup Synchronizer(see page 534) — 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.

Synchronizers are a now a dark feature, meaning they are hidden from users by default. To enable synchronizers, use the Structure Dark Features and Fine Tuning Interface(see page 499).

Sub-Tasks Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

The Sub-Tasks Synchronizer automatically places sub-tasks under their respective parent issues in the structure.

This synchronizer is available only when Sub-Tasks are enabled in your Jira instance and you have at least one Sub-task issue type defined.

Sub-Tasks Synchronizer Parameters

You can select which sub-task issue types the synchronizer works with. Issues of other issue types will not be affected.

This synchronizer only supports Import / Resync into Structure (more about resync(see page 520)).

Sub-Tasks Synchronizer Rules

- When there is a sub-task (of one of the selected types) and its parent issue is in the structure, the sub-task is also added to the structure and placed under its parent task.
- The parent issue must be in the structure already - the synchronizer does not add the parent AND sub-task, nor does it add a parent if the sub-tasks is already added.

  You can add parent issues to structure manually, or use a Saved Filter synchronization to add parent issues (and probably sub-tasks) automatically.
- If a sub-task is already in the structure, and is located under a different parent (or at the top level), it will be moved under its *subtask parent* (with all sub-issues that it may have).
- Changes in structure are not synced back to sub-tasks: if you place an issue under another issue, it will not become a sub-task.
  - If you move a sub-task away from its parent task, it will soon be moved back by the synchronizer.

Filter Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

The Filter Synchronizer automatically adds issues to a structure or removes issues from a structure based on a Saved Filter or a JQL query.This powerful synchronizer lets you control the contents of the structure with an issue filter (either a **Saved Filter** or an arbitrary **JQL Query**). It can be configured to automatically add or remove issues from a structure, or both.

Filter Synchronizer Parameters

Filter	This is where you determine the Saved Filter or JQL Query to sync with. Click **Select** to choose a saved filter or switch to **JQL query** and enter the JQL. ⚠ When this synchronizer is enabled and runs in background, it "listens" to Jira events about issues being changed. If the result of a query change without an issue being changed, the synchronizer will not detect the change and will not update the structure - for example, if you use JQL query `updatedDate > startOfMonth()`, the synchronizer will not update the structure at the beginning of a month, when the result of the query changes. You will need to do a Resync(see page 520) or use scheduled synchronization.
Add	Turns on **Add Mode**: the synchronizer will make sure that all issues from the filter's result are present in the structure.
Place added issue at the top level	If this is selected, the newly discovered issues from the filter result are placed at the top level, at the end of the structure.
Place added issue as a sub-issue of ...	Allows you to selecte an issue key (like PROJECT-123) of an issue that will serve as the parent for any newly discovered issues from the filter. These new issues will be placed as children of the selected issue, at the end of the current children list. ⚠ If the specified issue is not present in the structure, newly discovered issues will not be added to the structure.
Allow move	This option is only available if you select the "Place added issue as a sub-issue of..." option. This tells the synchronizer what to do if a matching issue is already added to the structure but located under a different parent issue. If the option is on, the synchronizer will move the issue (with all its possible sub-issues) under the specified parent issue. If the option is off, the issue will be left alone where it now resides.

Remove	Turns on **Remove Mode**: the synchronizer will remove issues from the structure when they no longer are present in the filter result. However, if an issue to be removed contains sub-issues that should stay in the structure, it will not be removed. **CAREFUL!** If Remove mode is enabled and another synchronizer is adding issues (for example, a Sub-tasks synchronizer), it could result in an endless cycle. Structure has safeguards in place to protect against this, but it is best practice to avoid this situation whenever possible. See Protection from Synchronizer Cycles(see page 522) for more details.
Remove only from where added issues are placed	Additional flag to remove issues only if they are either at the top level or under the issue where they were initially placed by the synchronizer. If this option is enabled, and you move an automatically added issue somewhere else, it will not be removed, even if it is no longer present in the search result.

This synchronizer only supports Import / Resync into Structure (more about resync(see page 520)).

If the Saved Filter used in the configuration is later deleted, or if you lose permissions to run it, the synchronizer will not work.

No matter how synchronizers are configured, they will only affect issues from the projects that are enabled for synchronization(see page 477).

Filter Synchronizer Rules

- The synchronizer adds issues from its filter's result to structure and/or removes issues that no longer are in the filter's result.
- Whenever an issue changes, a query is run to see if it matches the filter. On resync, all issues are checked.
- With **Add** mode on, an issue will be added to the structure if it matches the filter - even if the user has manually removed it from there. If the issue is already in the structure, it will not be affected, unless **Allow Move** is on - in which case it will be relocated under the specified parent issue.
- With **Remove** mode on, an issue will be removed from the structure if it does not match the filter - even if a user manually added it to the structure.
- When adding issues on Resync or Import, the synchronizer places them at the end of the structure (or under the specified parent issue), in the order that corresponds to the filter's order. However, if only part of the filter result is added (for example, because other issues are already in the structure), the final sequence of issues may be different from the filter result.

Automatic Branch Removal

> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

The “Double-check sub-issues” option is useful when you want to build and, more importantly, maintain a structure where you have:

- A certain set of issues on the top level
- All issues linked to those issues placed beneath them

Specifically, the Double-Check option is necessary for removing top level issues (and all the issues linked to them), when the top level issue no longer passes the synchronizer filter.

Example
You are trying to build a structure with all Open Stories on the top level and the issues that block them added below.

To build this structure you will need to configure the Filter synchroniser(see page 524), which will add Open Stories to the top level, and the Links synchroniser(see page 528), which will add linked issues.

To get the list of the top level issues, you could use the following JQL query:

```
issuetype = Story and status = open
```

However, if you use this as the filter query and select the remove option, the Filter Synchroniser will remove all children added by the Link Synchronizer, because the children do not pass the JQL query.

To solve this problem, you can extend the query with an S-JQL expression(see page 266), which returns both parents and their children, which are already in the structure - this will prevent the Filter Synchronizer from removing children from the Structure:

```
(issuetype = Story and status = Open) or issue in structure(“Open Stories Structure“,
"issueOrAncestor in [type = Story and status = Open]”)
```

Now the last step is to automatically remove any stories (and issues linked to them), when the status of the story changes. If the **Double-Check** option is not selected, once the Story status changes, the synchronizer will see that the Story should be removed, but will think that the children still pass the filter (because there were no explicit changes done to them). As a result, it will keep both the Story and the children in the Structure. Selecting the **Double-Check** option will force it to check if the children still pass the filter - and it will remove the whole branch.

Links Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

The Links Synchronizer maintains issue links between parent issues and child issues.You can use this synchronizer to replicate the hierarchy in the structure with issue links, or to import a hierarchy that was previously created with links.

The Links synchronizer is available only when Links are enabled and there is at least one link type.

This synchronizer supports Resync in both directions (Import and Export) (more about resync(see page 520)). Incremental synchronization watches both structure changes and issue link changes and applies those change to the other side (unless the Reverse contradicting changes(see page 529) option is specified).

No matter how synchronizers are configured, they will only affect issues from the projects that are enabled for synchronization(see page 477).

Synchronizers can add and remove Jira issue links based on the permissions of the user that installed the synchronizer.

Links Synchronizer Parameters

Link Type
The type of link to sync with. Links of other types will be ignored.

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

Parent Issue Filter and Sub-Issue Filter
If set, these filters determine which issues and links can be affected by the synchronizer:

- If a link's parent issue or sub-issue (as determined by **Link Direction**) doesn't pass the corresponding filter, then the link is ignored by the synchronizer. For example, if there are two issues that belong to the structure and pass the corresponding filters, but one of them falls out of its corresponding filter, the link will not be deleted.
- If there is a parent issue and a sub-issue in the structure, and either of them doesn't pass the corresponding filter, the synchronizer will not create a link between them.

You can use saved filters or JQL queries.

Scope
Defines which issues are affected by the synchronizer, based on whether they are in the structure or not.

- **Synchronize issues that are already in the structure** means that the synchronizer will affect only those issues that are already in the structure or reachable from it via issue links. Use this option when you need manual control over which of the linked issues appear in the structure.
    - If **Expand to sub-issues** is selected, the synchronizer will add sub-issues to the structure if their parent issue is in the structure.
    - If **Expand to parent issues** is selected, the synchronizer will add a parent issue to the structure if any of its sub-issues are in the structure.
- **Synchronize all issues that have links of selected type** means that the synchronizer will affect all issues that have matching issue links and pass the **Issue Filters**. For example, you can use this option to import all issue relationships represented by links into an empty structure.

This setting also controls which issue links can be deleted during export, manual resync *from* structure, or incremental synchronization. For example, when you remove a sub-issue from the structure, the synchronizer will remove the corresponding link only if it could have added this sub-issue back (that is, when either **Expand to sub-issues** or **Synchronize all issues** is selected).

> ⚠ **CAREFUL!** Please be careful when using this synchronizer with **Synchronize all issues** option selected, because Exporting or Resyncing *from* Structure would delete all the existing links of the selected type between issues that are not in the corresponding positions in the structure.

Removal
Defines how the synchronizer treats a sub-issue that doesn't have a link to justify its position in the structure (for example, when a user deletes the link from its parent issue):

- When **Move upwards** is selected, the synchronizer will move such an issue up the hierarchy until it's either at the top level of the structure or in a position that doesn't contradict the settings (for example, under an issue that does not pass the **Parent Issue Filter**).
- When **Remove** is selected, the synchronizer will remove such an issue from the structure, together with all its sub-issues.

Primacy
By default, when a synchronizer is installed and enabled, it tracks changes made by users and applies them to the "other side":

- When a user creates or deletes issue links, the synchronizer adjusts the structure accordingly.
- When a user changes the structure, the synchronizer creates or removes the corresponding links.

You can use the **Reverse contradicting changes** option to override this behavior and specify the primary place where issue relationships are stored:

- With **Structure primacy**, when a user creates or deletes a link that is within the scope of the synchronizer but contradicts the structure, that change will be reverted. One needs to change the structure to adjust issue relationships.
- With **Links primacy**, the synchronizer reverts changes to the structure that contradict issue links. One needs to change the links to adjust an issue's position within the structure. Note that this does not apply to reordering issues without changing their parents.

> ⚠ Primacy does not apply during Export, Import or manual Resync.

Links Synchronizer Preserves Links Between an Added List of Issues

When a list of 2 or more issues is added to the structure, links between these issues are preserved, and the issues form a hierarchy according to these links. Such a situation may arise, for example, when moving several issues into the structure at one time.

This differs from the default behavior when the **Reverse contradicting changes** option is not selected: normally, if an issue *A* is added to the structure as a sub-issue *B*, and both of them pass the Issue Filters, Links synchronizer would establish a link between *A* and *B* and remove all other links of the corresponding type where *B* is on the sub-issue end of the link. When a list of issues is added, however, the synchronizer behaves as if **Links primacy** was selected.

Links Synchronizer Rules

- When the synchronizer is enabled:
  - Changes in the structure will be reflected by creating and removing links of the selected type.
  - Links created or removed by the user will be automatically reflected in the structure.
- Links created and removed by the synchronizer are not recorded in the issue history, and the issue update time is not changed (due to performance reasons).
- Use Resync (*from* Structure to Links) or Export to update the links according to the structure.
  - If **Synchronize all issues** is selected, all other links of the selected type will be deleted.
  - Otherwise, the links that are reachable from the structure considering **Expand to...** options, but not represented in the structure, will be deleted.
- Use Resync (from Links *into* Structure) or Import to add and rearrange the issues in the structure according to the existing links.
  - If **Synchronize all issues** is selected, all issues with matching issue links will be added to the structure.
  - Otherwise, only the issues reachable from the structure considering **Expand to...** options will be added.
- Links that violate hierarchy restrictions are treated as follows:
  - If a sub issue has more than one parent issue, the most recent issue link is used.
  - If there is a sub-issue cycle, the oldest issue link is not used.
  - There is an exception to the two preceding rules: Links synchronizer prefers to use links between an added list of issues(see page 530), 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[103] describes how to add JIRA startup properties.
2. Restart Jira

103 https://confluence.atlassian.com/display/JIRA/Setting+Properties+and+Options+on+Startup

Agile Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

The Agile Synchronizer lets you synchronize the position of issues in the structure and on an Agile board (such as a Scrum or Kanban board) using Rank synchronization, and synchronize Epic fields with the position of stories under epics in the structure.

Agile Synchronizer Parameters

Synchronize	Choose the mode of operation: • Use Agile Board configuration • Use custom projects and fields configuration
	Agile Board mode parameters:
Agile Board	Select the Jira Agile board to synchronize with. The issues matching the Board query will be synchronized. If the structure contains other issues, they will not be affected.
Synchronize Rank	If checked, repositioning issues in the structure will also change their Rank respectively. Changing Rank in Jira will likewise cause the issue to be repositioned in the structure.
Synchronize Epics	If checked, placing an issue under an epic in the structure will add it to that epic in Jira. Similarly, if you add an issue to an epic (on the Scrum board or by changing Epic/Theme field of the issue), the issue will be repositioned under that epic in the structure.
Auto-add sub-tasks	If checked, all sub-tasks of an issue will be added to the structure as well and positioned under the parent issue.
	Custom projects and fields parameters:
Projects	Select the Jira project(s) to synchronize with. If the structure contains issues from other projects, they will not be affected. The issues from all selected projects will be synchronized using the same Global Rank field.
Rank Field	The field of type "Rank" that holds the rank (backlog order) for the selected Project. If you do not wish to synchronize rank, select *Don't synchronize*.

Synchronize Epics	If checked, placing an issue under an epic in the structure will add it to that epic in Jira. Similarly, if you add an issue to an epic (on the Scrum board or by changing Epic/Theme field of the issue), the issue will be repositioned under that epic in the structure.
Auto-add Subtasks	If checked, all sub-tasks of an issue will be added to the structure as well and positioned under the parent issue.

This synchronizer supports both Import and Export / Resync into/from Structure (more about resync(see page 520)). Incremental synchronization watches both structure changes and Jira changes and applies the change to the other side.

> ⚠ **CAREFUL!** Please be careful when using this synchronizer, especially when you add multiple issues to the Structure, as this may lead to massive updates in the Agile ranks. There is no "undo" option.

Agile Synchronizer Rules

Common Rules

- Issues that do not belong to the synchronized project(s) are not affected.
- This synchronizer does not add issues to the structure (with two exceptions, explained below). You can use a Saved Filter synchronizer(see page 524) together with an Agile synchronizer to automatically add and position issues.

Sub-Tasks Synchronization

- With the **Auto-add sub-tasks** mode on, sub-tasks are added to the structure if their parent is in the structure.
- The sub-tasks are forced to stay under their parent. If you move a sub-task somewhere else, it will jump back under the parent again. You can rearrange the order of the sub-tasks, which will be sync'ed to the Agile Rank, if the Rank Field is configured.

Rank Synchronization

- Repositioning issues in the structure causes a Rank change and the repositioning of issues on the Planning Board.
- Rearranging issues on the Planning Board causes the issues to be rearranged in the structure.
- When issues are repositioned in the structure according to Rank, they are never moved under a different parent issue.

  > ⚠ This restricts the possible rank changes in Jira - you can only move an issue to the position of another issue that is under the same parent issue in the structure; otherwise, the issue will "jump back" later.

Epic Synchronization

- Placing an issue under an epic in the structure will cause its Epic field to change to that epic.
  - It does not matter what depth the issue is at in the structure. A sub-sub-sub-issue of an epic issue will still have its Epic field updated.
- If you move an issue in the structure so that it is not under any epic, its Epic field will be cleared.
- If you manually change the Epic field to point to a different epic, the issue will be repositioned under that epic in the structure.

    - An issue that has the Epic field pointing to an epic that is in the structure will be automatically added to the structure.
- If you clear the Epic field or change it to point to an epic that is not in the structure, the issue will be moved up in the structure until it is no longer under any epic.

How to Add Issues to Structure Sync'ed with JIRA Agile
When the Agile synchronizer is enabled, it automatically updates the Agile order in background when any Structure change happens. This means that if you carelessly add issues from the sync'ed project to the structure in some random order, their ranks will be updated according to that order.

To add issues to the structure without breaking the existing backlog order:

- If adding manually on the Structure Widget, use a JQL search and add the *order by Rank* clause to the end of the query, using the rank field that is used by the synchronizer.
- Select the position of the added issues carefully (using drag-and-drop or copy/paste) - the order is likely to change unless you place issues under another issue without any other sub-issues (see *Syncing Partial Orders* below).
- If using a Saved Filter synchronizer to add issues, add the *order by Rank* clause to the Saved Filter's query. Note that any issues added by the Saved Filter synchronizer will appear at the end of the structure, and have the lowest ranking.

Syncing Partial Orders
Agile Boards are flat (except for sub-tasks), and the Structure is hierarchical - so it is not possible to precisely rearrange a structure to have all issues come in the same order as they do on the Planning Board, without changing issue parents or making the Structure also flat.

To that end, the structure syncs subsets of the issues in the hierarchy with Agile Rank. For example, consider the following Structure:

A	
	B
	C
D	
	E
	F

It is not possible to rearrange the sub-issues so they come in the following order: B, E, C, F - although this is possible on the Planning Board. Instead, the structure will synchronize sub-sets of the issues in the Structure with Jira. The following sub-sets will be synchronized separately:

- A, D - top-level issues: A must come before D on the Planning Board
- B, C - sub-issues of A are sync'ed separately, so B must come before C on the Planning Board
- E, F - sub-issues of D are sync'ed separately, so E must come before F on the Planning Board

In Jira version 6.1 and later, the epics are treated as a separate set of issues, different from Stories and other non-Epics. To accommodate this, Structure updates the rank of issues using a "partial order" approach, syncing epics and non-epics separately. This means that if an epic comes before a Story on the Structure Board, it is not required that they come in the same order on the Scrum Board.

Status Rollup Synchronizer

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

The Status Rollup synchronizer automatically aggregates statuses of the sub-issues and updates the status of the parent issue. For example, if all sub-issues are *Resolved,* it can update the parent issue's status to *Resolved*.

Status Rollup Synchronizer Parameters

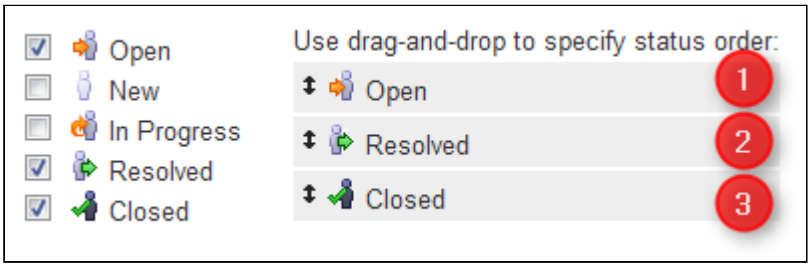
Enabled Projects	Only issues belonging to the selected projects are **changed**. It does not matter what project sub-issues belong to, as long as their parent belongs to the enabled project — every sub-issue counts with its status.
Enabled Issue Types	Determines which types of issues can be changed by the synchronizer. As with the enabled projects, only the parent issue type is checked.
Statuses Rolled Up	The selection and order of statuses that are used to calculate the parent issue status. Parent issue status is set to the *earliest* status among its sub-issues. If a sub-issue has a status not selected in this parameter, the parent issue is not changed.
Allowed Transitions	For every status, you can select which transitions the synchronizer can make to move an issue to that status.
Resolution	The value the *Resolution* field is set to when a workflow transition requires it. By default, a current or default value for Resolution is used.

The synchronizer is normally installed, resynced and used in the Incremental mode, tracking changes to issues and the structure and updating issues based on those changes. The synchronizer supports Exporting from Structure, changing statuses of the issues in the structure on a one-time basis.

How the Status Rollup Synchronizer Works

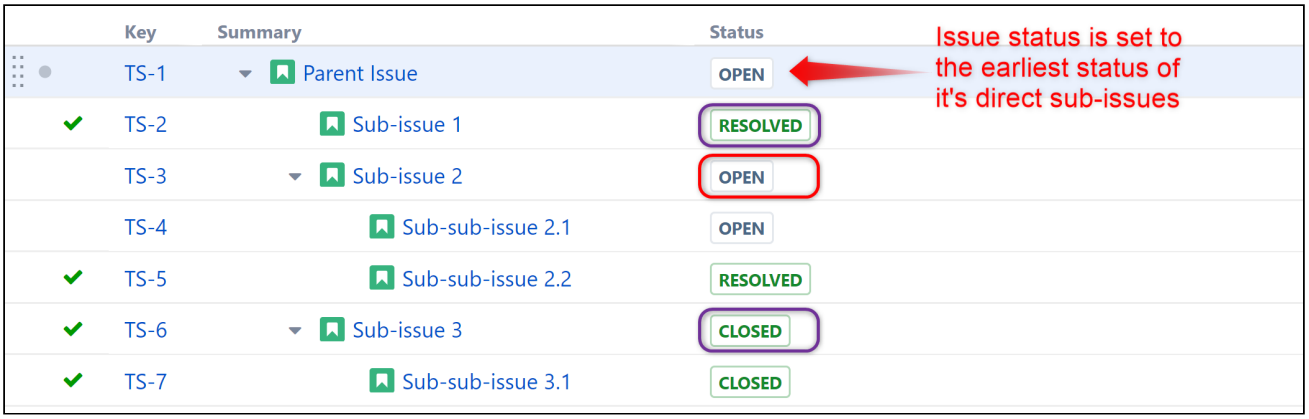
The synchronizer tracks updates to issues and to the structure, and then tries to make sure that the status of the parent issue corresponds to the aggregate status of its direct children.

When you configure Status Rollup, the most important parameter is the selected Statuses and their order:



Statuses that are not selected in the parameters are not recognized by the synchronizer. If a sub-issue has one of the unselected statuses, the synchronizer does not change the parent issue.

The order of the selected statuses should correspond to *earliest-to-latest* order of phases of the workflow. For example, the screenshot above shows a configuration where *Open* is followed by *Resolved*, which is followed by *Closed*. With that configuration, once all sub-issues of an issue are *Resolved*, the synchronizer will try to make the issue *Resolved* too. Once all sub-issues are *Closed*, the issue will be made *Closed*. But if at least one sub-issue happens to be *Open*, the issue status will be set to *Open* — because it is the earliest status in the specified order.



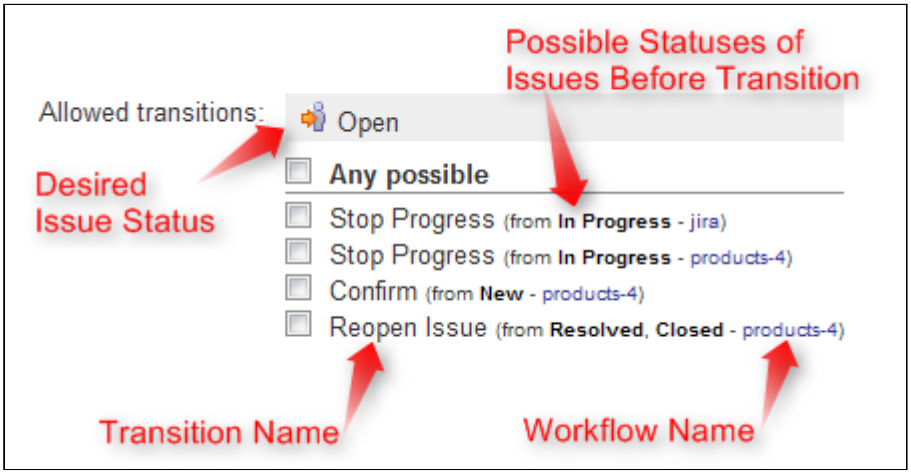
On the screenshot above:

- **Sub-issue 3** has a single sub-issue, which has status **Closed** — since all of its sub-issues are closed, Sub-issue 3 should be **Closed** too.
- **Sub-issue 2** has one **Open** sub-issue and one **Resolved** sub-issue — it should be **Open**, because Open status comes before Resolved in the order specified earlier.
- **Parent Issue** has sub-issues that have statuses **Open**, **Resolved** and **Closed** — so it should be **Open** for the same reason. Once Sub-issue 2 is **Resolved**, Parent Issue will be automatically **Resolved** (because that will be the new earliest status). Once all sub-issues are **Closed**, Parent Issue will automatically be **Closed**.
- **Sub-issue 1**, **Sub-sub-issue 2.1** and **Sub-sub-issue 2.2** do not have sub-issues of their own, so the synchronizer does not change their status.

If one of the sub-issues gets a status not listed in the synchronizer configuration, the synchronizer just skips the issue. For example, if we change the status of **Sub-issue 2** above to **In Progress**, **Parent Issue** will not be updated. If we then change the status of **Sub-issue 2** to **Resolved**, **Parent Issue** status will be updated to **Resolved**.

How Status is Changed

Jira allows a status to be changed only through a workflow transition, so the only way the Status Rollup synchronizer can set the desired status on an issue is to apply a workflow transition. Therefore, when you select a status, you also need to select which transitions the synchronizer is allowed to make.



The synchronizer:

1. Checks what status the issue currently has;
2. Calculates what status it should have, based on the statuses of sub-issues;
3. Finds workflow transitions that can transfer the issue from the current status to the required status;
4. Checks which of those transitions are allowed by the configuration; and
5. Tries to apply matching transition number one. If that fails, it tries the next, and so on.

Note that all transitions are done under the account of the user who installed the synchronizer.

Why Can a Workflow Transition Fail

It's not guaranteed that the synchronizer will be able to change the Status, because workflows are too flexible, and there are many reasons that a given transition which you have allowed in the configuration can fail to execute. Here are some of the possible causes:

- You (the user who has installed the synchronizer) do not have the required permissions to make the transition
- You are not the Assignee of the issue — required for In Progress status
- Some other pre-condition defined in the workflow fails
- The workflow transition requires a field to be set on an issue that has no default value

As described above, it's possible that there are several possible transitions from one status to another. The synchronizer will try all of them unless one of them succeeds.

If the synchronizer fails to update the status, a warning message will be written into the server logs (subject to logging configuration).

Changing Resolution

You can set up a specific *Resolution* value to be set whenever a transition involves changing the resolution. If you don't specify this parameter, the default resolution or already existing resolution will be used.

In order to tell which issues have been automatically moved to a status like Resolved or Closed, you can set up a special resolution such as *Auto-Resolved*.

Manually Changing Status of an Issue That Has Sub-Issues

Even if an issue has sub-issues and is subject to Status Rollup, you can manually change its status. Although the synchronizer will **not** be forced to recalculate the status of that issue immediately, it will recalculate the status if any of the sub-issues change – probably reversing your change - if it finds an allowed transition.

If you'd like the synchronizer to only move issues *forward* (for example, from *Open* to *Resolved*), but not vice versa, you can configure the allowed transitions accordingly.

5.15.5.11 Undo Synchronizer Actions

Dark Feature - Synchronizers are hidden by default

We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.

To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

Caution should always be exercised when using synchronizers. An incorrectly configured synchronizer or an accidental move can result in unexpected changes to both a structure and Jira.

Fortunately, Structure provides a method for undoing changes made by synchronizers.

Synchronizer Audit Log

To review actions that were completed automatically by installed synchronizers:

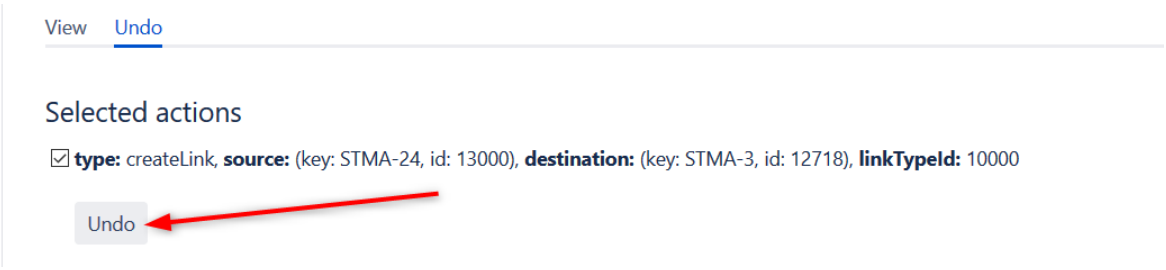
1. Go to the Administration menu and select **Structure | Support**.
2. Under Structure Support, locate the Synchronizer Audit Log section and click **View Synchronizer Audit Log**.
3. On the Synchronizer Audit Log screen, scroll down to view a list of recent synchronizer actions. If you do not see the action(s) you wish to undo, you can search the audit log based on timeframe, Sync Instance ID and Structure ID.

If you do not have access to the Administration menu, speak with your Jira administrator.

Undo Synchronizer Actions

Once you have identified the action(s) you wish to undo, select them within the Audit Log and click **Undo**.

On the next screen, review and confirm your selection by clicking **Undo**.



5.15.5.12 Copying Synchronizers

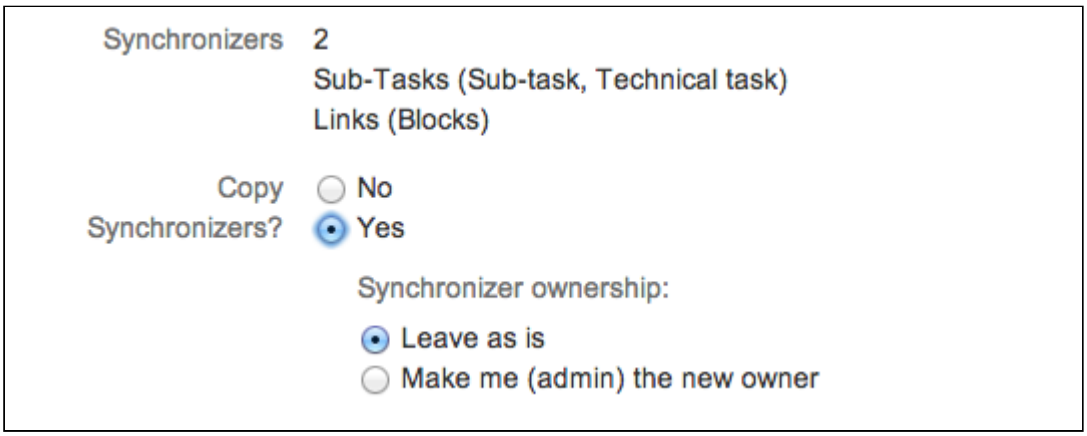
> **Dark Feature - Synchronizers are hidden by default**
>
> We recommend using Automation (see page 122)instead of Synchronizers. Automation is safer, provides additional functionality and flexibility, and uses fewer system resources.
>
> To enable synchronizers, see Advanced Configuration and Dark Features(see page 499).

When copying a structure[104] 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.

104 https://wiki.almworks.com/display/structuremaster/Copying+a+Structure

Copying Parameters



You can decide to leave the original ownership of a synchronizer ("**Run As**" parameter) or make yourself a new owner for each of the copied synchronizers.

⚠ Only Jira administrators can change the synchronizer ownership.

⚠ Making yourself the new owner means that all synchronizers in the copied structure will run under your account.

Required Permissions

To be able to copy synchronizers, you need permissions to create and configure synchronizers(see page 522).

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

5.16 System Requirements

5.16.1 Atlassian Platform

Jira Versions Supported	7.13 – 8.14 (by the latest version) See also: Platforms supported by Jira[105]

105 https://confluence.atlassian.com/adminjiraserver071/supported-platforms-802592168.html

Jira Editions Supported	Jira Core, Jira Software, Jira Service Management (formerly Jira Service Desk)
Jira Data Center	Supported See Server Requirements below
Confluence Versions (Structure.Pages)	6.13 – 7.9 (by the latest version)

5.16.2 Databases

Databases used by Jira are also supported by Structure.

5.16.3 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	

5.16.4 Server Requirements

- At least 100MB of free disk space is needed on the server. See Structure Files Location(see page 498) for details.
    - On Jira Data Center, each node must have sufficient free disk space in the local home.
- Java process running Jira needs at least an additional 200 MB of heap memory. If running on Java 7 or earlier, ensuring sufficient free PermGen space is recommended. See Memory Guidelines(see page 469) for details.
- Jira process must have read/write permissions to the Jira (local) home directory to create the `structure` sub-directory automatically.

5.16.5 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.

5.17 Confluence Gadget - Admin Configuration

The following articles will assist admins with enabling and configuring the Structure gadget within Confluence.

- Adding Structure Gadget to Confluence Configuration(see page 541)
- Setting Up CORS Filter in JIRA(see page 542)
    - Nginx Configuration Option(see page 542)

5.17.1 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[106]
    a. Use **OAuth Authentication** to let the Confluence page viewer authenticate separately with Jira. (Preferred)
       Documentation: Configuring OAuth for an Application Link[107]
    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[108]

       > 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[109]
3. Check the sample page to confirm whether you can include the Structure macro and get data from Jira.

106 https://confluence.atlassian.com/doc/linking-to-another-application-360677690.html
107 http://confluence.atlassian.com/display/APPLINKS/Configuring+OAuth+Authentication+for+an+Application+Link
108 http://confluence.atlassian.com/display/DOC/Configuring+Trusted+Applications+Authentication+for+an+Application+Link
109 https://confluence.atlassian.com/display/DOC/Registering+External+Gadgets

5.17.1.1 Troubleshooting

If you have problems using the Structure gadget in Confluence, check the browser's console. If you see errors saying that loading some of the resources is denied, it is likely due to a CORS problem in Jira. To work around that problem, see Setting Up CORS Filter in JIRA(see page 542).

5.17.2 Setting Up CORS Filter in JIRA

Sometimes Structure Gadget fails to load correctly in Confluence. You might see missing icons or the application can fail to work.

This may happen because of a known Jira issue that prevents the Structure gadget from loading resources from Jira when it's being served in Confluence on another web domain.

To work around this problem, you can set up a CORS filter in the Tomcat server that runs Jira (Nginx users may want to consider this alternative Nginx Configuration Option(see page 542)):

1. Copy cors-filter-2.4.jar, java-property-utils-1.9.1.jar from CORS docs[110] 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

5.17.2.1 Nginx Configuration Option

Some Nginx proxy users reported that adding the following block directive is an effective workaround for addressing the missing CORS headers issue.

Please replace ***XXXX*** with your actual Confluence domain name:

110 http://software.dzhuvinov.com/cors-filter-installation.html

```
location ~* \.(eot|ttf|woff|woff2)$ {
    add_header Access-Control-Allow-Origin "https://confluence.XXXX.com";
    try_files $uri @jira;
  }
  location / {
    try_files $uri @jira;
  }

  location @jira {
    proxy_pass http://localhost:8080 ;
    proxy_read_timeout 180s;
    proxy_http_version 1.1;
    proxy_redirect off;
  }
```

5.18 Clearing Structure Caches

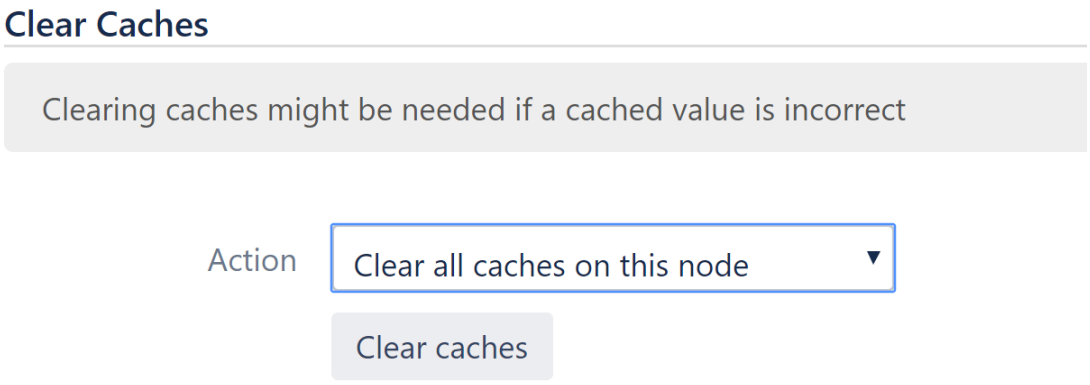
Structure caches can be cleared for specific users or the whole Jira instance. In Jira Data Center, it's possible to clear caches for the current node or all nodes in the cluster.

> ⚠ Jira's own caches are not affected by this action, only Structure caches are cleared.

There are two ways to clear Structure caches:

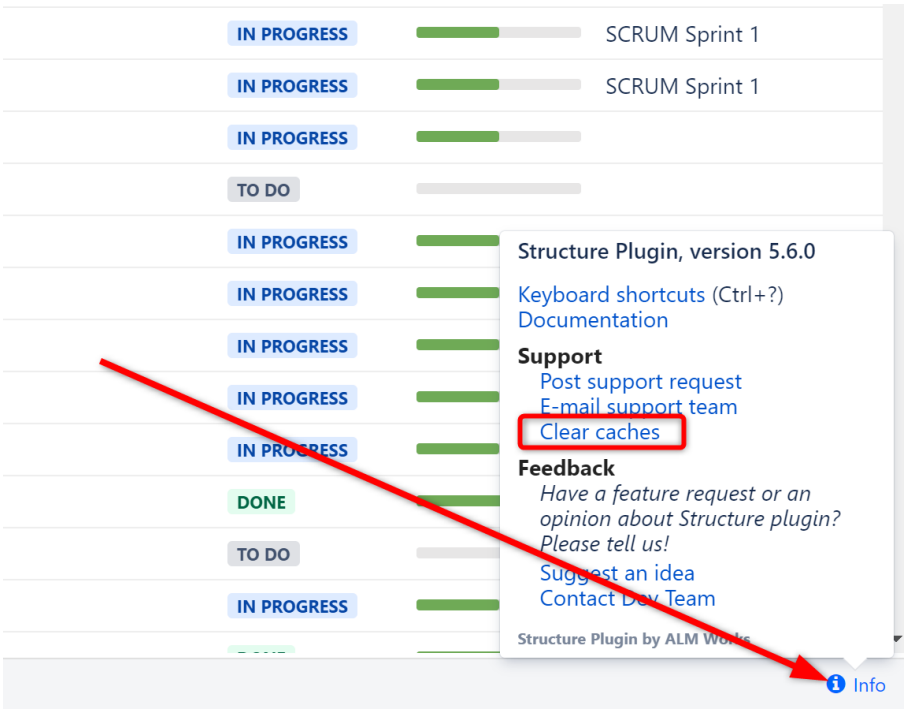
Admin-only Method

1. Navigate to **Administration | Structure | Support**
2. Scroll down to the **Clear Caches** section and select the caches to be cleared.
3. Click **Clear Caches**.



All-user Method

The Clear caches option is also available to admins and Structure users via the Info pop-up in the bottom right corner of the Structure panel.

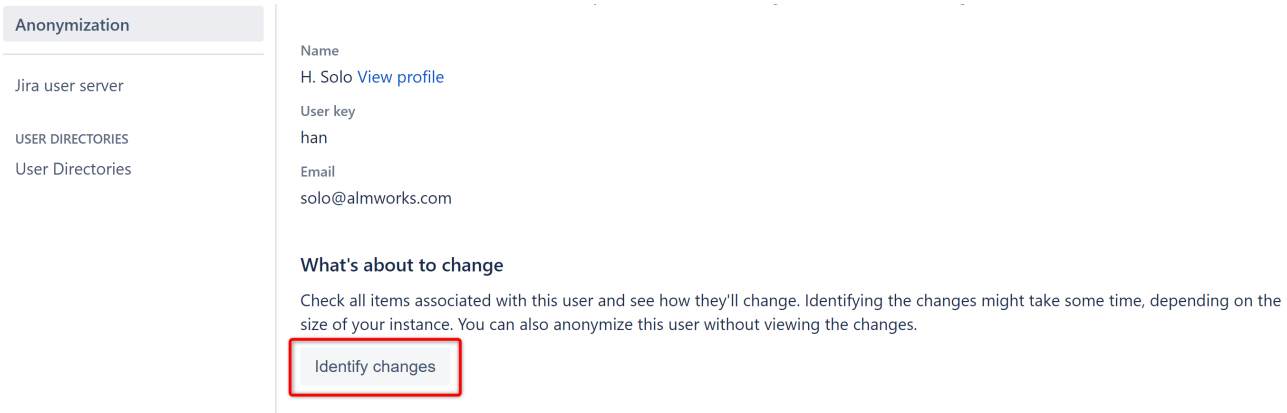


5.19 Anonymizing Users

When you Anonymize a user[111] 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.

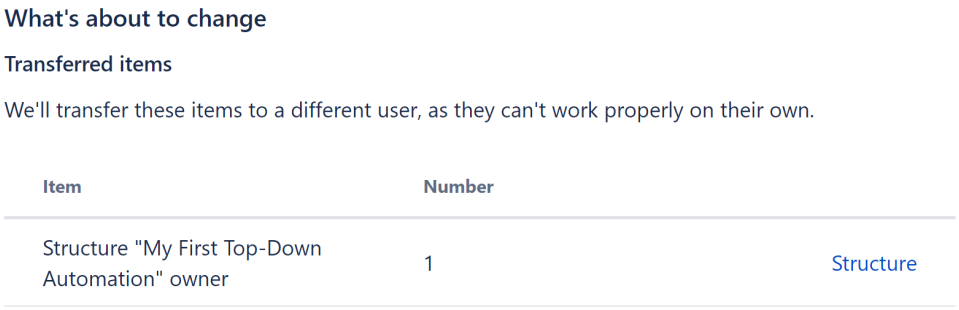
5.19.1 Review User's Structures

To review the structures and/or views owned by the user, on the Anonymization screen, click **Identify changes**.

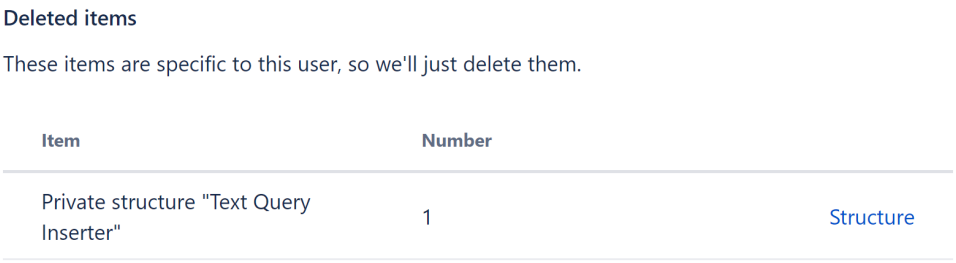


Any shared structures or views will appear under the **Transferred items** list. These will be transferred to a new user after the anonymization.

111 https://confluence.atlassian.com/adminjiraserver/anonymizing-users-992677655.html



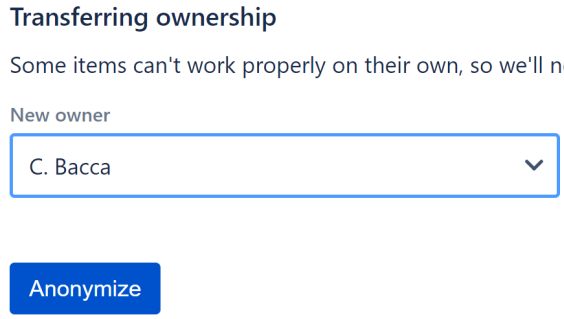
Any private structures or views (not visible to anyone but the user) will appear under the **Deleted items** list. These will be deleted.



To review any of the structures on these lists, click the **Structure** link to the right.

5.19.2 Transfer Ownership

Ownership for any shared structures or views will be transferred to the user you select in the **Transferring ownership** section.



5.20 Structure Troubleshooting

- Collecting Support Zip(see page 546)
- HAR Network Report(see page 546)
- Troubleshooting Synchronizers(see page 547)
- Structured JQL Troubleshooting(see page 549)
- Collecting Performance Snapshots(see page 549)
- Sending Files to Support Team(see page 555)

- Troubleshooting Performance and Stability Issues(see page 556)

5.20.1 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.

5.20.2 HAR Network Report

HAR Network Report is something we (ALM Works Support) may ask you to collect, to help us understand a tricky problem that we could not reproduce.

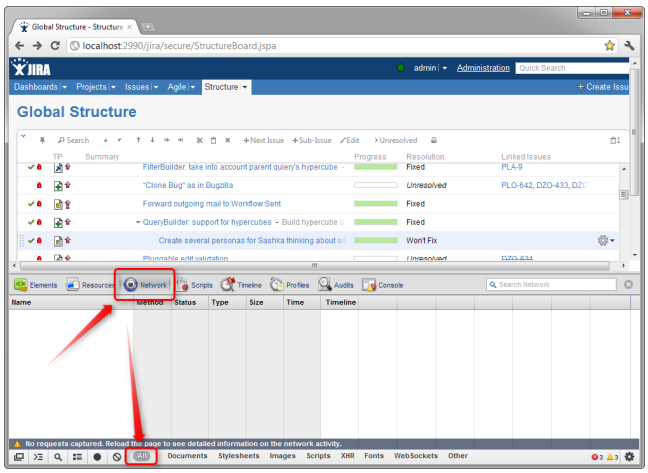
> HAR stands for HTTP Archive Format[112], a text-based format for the log of network communications between a user agent (the browser) and a web server. You can also use this report with a HAR Viewer[113] for in-depth analysis of your JIRA page load performance. (Be aware though that with an online viewer you may transfer sensitive or security-related information to a third party.)

5.20.2.1 Collecting HAR Report with Google Chrome

1. Open a new Chrome window and navigate to the page where the problem happens.

112 http://www.softwareishard.com/blog/har-12-spec/
113 http://www.softwareishard.com/blog/har-viewer/

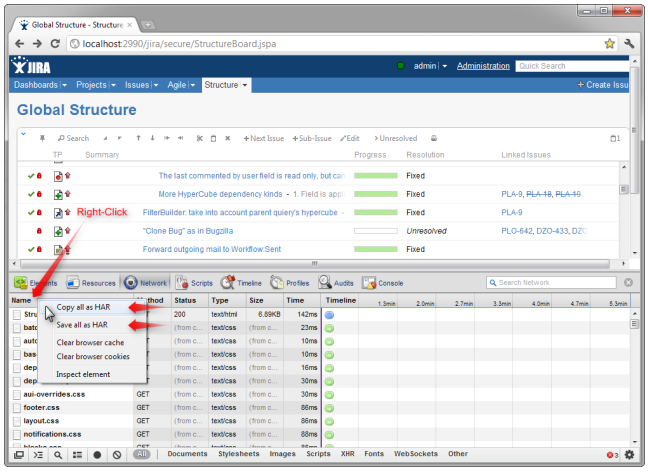
2. Press **Ctrl+Shift+I** or use menu **Wrench | Tools | Developer Tools** to open a section with developer tools. Switch to the **Network** tab there. Make sure **All** tab is selected below.



3. Reload the page by using **Ctrl+R** or clicking the Reload button. This will make Network tab log all network exchange during page load.

> The network tab will start collecting information or network exchange automatically after it's opened. If you know that the problem is not related to the initial page load, you may skip this step to avoid adding extra data to the log. If unsure, reload the page to collect the full report.

4. Reproduce the problem being analyzed.
5. After the problem has been reproduced, **right-click** on the **Name** column in the Network tab and choose either **Save all as HAR** or **Copy all as HAR**



6. Paste the report into an e-mail to our support, or attach the saved .HAR file.

5.20.3 Troubleshooting Synchronizers

Structure synchronizers(see page 514) 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.

5.20.3.1 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.

5.20.3.2 Log Files

To get detailed reports about what's going on, you can reconfigure your JIRA logging so that structure synchronizers can produce more verbose messages. Also, you might want to direct messages from the synchronizers into separate log files.

The appearance of detailed synchronizer messages is governed by the log level: the lower the log level, the more detailed messages can appear. By default, log level for structure synchronizers is `WARN`, and you can set it to lower levels, like `DEBUG` (the lowest one.) You can set the logging level either temporarily(see page 548) (until the next JIRA restart) or permanently(see page 548).

To see the list of possible log levels and other general information regarding logging in JIRA, please refer to JIRA logging documentation[114].

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[115] 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[116].

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:

114 https://confluence.atlassian.com/display/JIRA/Logging+and+Profiling
115 https://confluence.atlassian.com/display/JIRA/Managing+Global+Permissions
116 https://confluence.atlassian.com/display/JIRA/JIRA+Installation+Directory

```
log4j.appender.structure-sync=com.atlassian.jira.logging.JiraHomeAppender
log4j.appender.structure-sync.File=structure-sync.log
log4j.appender.structure-sync.Threshold=TRACE
log4j.appender.structure-sync.MaxFileSize=20480KB
log4j.appender.structure-sync.MaxBackupIndex=1
log4j.appender.structure-sync.layout=org.apache.log4j.PatternLayout
log4j.appender.structure-sync.layout.ConversionPattern=%d %t %p %X{jira.username}
[%c{4}] %m%n

log4j.logger.com.almworks.jira.structure = DEBUG, structure-sync, console
log4j.additivity.com.almworks.jira.structure = false
```

5.20.4 Structured JQL Troubleshooting

If a Structured JQL(see page 266) 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[117] or write to support@almworks.com[118] 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[119] — 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[120].
   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.

5.20.5 Collecting Performance Snapshots

Performance snapshots allow ALM Works support team to analyze performance-related problems on your JIRA server without direct access to it.

5.20.5.1 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).

117 https://answers.atlassian.com
118 mailto:support@almworks.com
119 https://confluence.atlassian.com/display/JIRA/Search+Indexing
120 https://confluence.atlassian.com/display/JIRA/Configuring+JIRA+Options

File	Modified
atlas-yourkit-0.2.jar[121] Atlas-YourKit, profiling plugin for JIRA	Apr 05, 2017 by Roman Gagarskiy[122]

5.20.5.2 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[123] for details.

121 https://wiki.almworks.com/download/attachments/32223442/atlas-yourkit-0.2.jar?api=v2
122 https://wiki.almworks.com/display/~rg
123 http://yourkit.com/docs/10/help/agent.jsp

5.20.5.3 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.

5.20.5.4 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.

5.20.5.5 Sending the Snapshots to Support Team

By default, snapshots are written into `<jira_home>/yourkit/snapshots` directory. Locate it and create a ZIP archive of all relevant snapshot files.

Please send the ZIP to us as described here: Sending Files to Support Team(see page 555).

5.20.5.6 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.)

5.20.5.7 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[124]. 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.

124 http://yourkit.com/

Download Profiling Agent

Download the ZIP with profiling agent from here: jira-profiler-v1-yjp956.zip[125] `md5sum` `e3ea2b72ef4b22584c641425275050d0`

Unpack the downloaded ZIP file into the directory where you have JIRA installed (**not** JIRA home!). This will create `<jira_install>/profiler` directory under your JIRA installation path.

> You can unpack the profiler into any other directory, but this instructions and our scripts assume that the profiler is unpacked into JIRA install dir.

If you will be able to restart JIRA before profiling, this is all you need — you can proceed to restarting JIRA with Profiling(see page 552).

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):

---

125 http://almworks.com/.files/jira-profiler-v1-yjp956.zip

```
set JVM_SUPPORT_RECOMMENDED_ARGS=-agentlib:
%~dp0..\profiler\bin\win64\yjpagent=port=10001
,onlylocal,dir=%~dp0..\profiler\snapshots,delay=20000 -XX:MaxPermSize=500m
```

b. On other OS, set or append the following parameters to `JVM_SUPPORT_RECOMMENDED_ARGS` in `<jira_install>/bin/setenv.sh` (following is a single long line):

```
JVM_SUPPORT_RECOMMENDED_ARGS="-agentpath:`dirname \"$0\"`/../profiler/bin/
linux-x86-64/libyjpagent.so=port=10001,onlylocal,dir=`dirname \"$0\"`/../
profiler/snapshots,delay=20000 -XX:MaxPermSize=500m"
```

3. Note that in the lines above, you should change **win64** or **linux-x86-64** to the name of the directory where the correct profiler agent for your OS/Java is located.
4. You may also need to change **port=10001** to make profiling agent listen on some other TCP port - in case port 10001 is already taken.
5. Stop JIRA and start it again.
6. Watch `<jira_install>/logs/catalina.out` for YourKit message like *[YourKit Java Profiler 9.5.6] Loaded.*

Use Copy & Paste to copy the parameters and then edit them in the setenv.sh

If the parameters are set incorrectly, JIRA start may fail. Verify that you have specified the agent directory correctly. Also verify that `<jira_install>` directory path does not contain spaces.

Profiler agent will use directory `<jira_install>/profiler/snapshots` to write performance snapshots - it must be write-accessible to the JIRA process.

Now you can proceed to #Running Profiling Session(see page 554).

Attach Profiler Agent to JIRA without Restarting

If you have restarted JIRA with profiling, skip this step.

If possible, restart JIRA with profiling instead of attaching profiler agent on the fly.

You will need the full distribution of YourKit downloaded at step 1.1. You will need to run a Java program as specified below - with the same version of Java that JIRA runs on. We assume that it is in your PATH variable in the command-line, but if it's not - you need to specify a full path to `java`.

1. Find out the process ID of the process that runs JIRA. You can use `jps` command from the Java distribution.
2. Find out the location of JDK (Java Development Kit). If you don't have JDK installed (only JRE), this procedure won't work. Typically JDK home is stored in the command-line environment variable JAVA_HOME.
3. Change current directory to `<jira_install>/profiler/yjp-9.5.6`. (You may have a different version of yjp.)
4. Run the following command, substituting JIRA process ID instead of ***PID***.

a. On Windows:

```
java -cp lib\yjp.jar;%JAVA_HOME%\lib\tools.jar com.yourkit.Main -attach
PID port=10001,onlylocal,dir=<jira_install>\profiler\snapshots
```

Replace <jira_install> with the full path of the JIRA installation folder.

b. On other OS:

```
java -cp lib/yjp.jar:$JAVA_HOME/lib/tools.jar com.yourkit.Main -attach PID
port=10001,onlylocal,dir=`pwd`/../snapshots
```

The command should output something like this:

```
Attaching to process 60108 using options port=10001,onlylocal,dir=..\snapshots
 The profiler agent has attached. Waiting while it initializes...
 The agent is loaded and is listening on port 10001.
 You can connect to it from the profiler UI.
```

Running Profiling Session

To successfully run a profiling session, you need to have JIRA running with a profiling agent, as explained above. The agent does not add much overhead when being idle — it sits there waiting for your commands to start a profiling session.

General Procedure

The profiling session is controlled by sending commands to the profiling agent (within the JIRA process). The program that is used to send the commands is `yjp-controller-api-redist.jar`, located in `<jira_install>/profiler`. The common format for running this program is:

```
java -jar yjp-controller-api-redist.jar localhost 10001 <command>
```

The <*command*> is replaced with some actual command, and if you changed the default port of the agent from 10001 to something else, you need to specify that port number here instead of 10001. This command should be run from `<jira_install>/profiler` directory.

We are assuming that `java` is on your PATH. If not the case, use the full path to `java` executable.

CPU Performance Analysis

If JIRA is unresponsive or burns CPU extensively, you can run CPU analysis session.

1. Start session with the following command:

```
java -jar yjp-controller-api-redist.jar localhost 10001 start-cpu-sampling
```

2. Let JIRA work for some time. If needed, take a specific action that causes the problem to manifest.
3. Stop session and record a snapshot:

```
java -jar yjp-controller-api-redist.jar localhost 10001 capture-performance-
snapshot
```

Sending the Snapshots to Support Team

By default, snapshots are written into `<jira_install>/profiler/snapshots` directory. Locate it and create a ZIP archive of all relevant snapshot files. If the ZIP is less than 10 Megabytes, it's ok to send it to us by e-mail.

If the ZIPPed snapshot is 10 MB or larger, you need to use FTP to send it over to us:

1. Use any FTP client (`ftp` or `lftp` from the command line).
2. Connect to host `f.almworks.com`
3. Use login name `almftp` and password `almftp`
4. Upload files to the root folder.
5. After the upload is finished, please send us an e-mail with a notification that you have uploaded the snapshots.

> You will not be able to list or download files from that FTP, and your FTP client may show errors about that. That's ok and should not prevent you from uploading snapshots.

After Profiling Session

You may want to continue running JIRA with the profiling agent loaded, since it does not product much overhead. Make sure you have stopped all the monitoring.

For a safer / cleaner environment, you can restart JIRA with the profiling options in `setenv` script commented out.

5.20.6 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).

5.20.6.1 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[126], please use Service Desk to upload and attach the files to the ticket. Size limit is 20 MB per upload.

5.20.6.2 Send Files by E-mail

File size limit: 20 MB

126 https://support.almworks.com/servicedesk/customer/portal/8

You can send the files to support@almworks.com[127]. 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.

5.20.6.3 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.

5.20.7 Troubleshooting Performance and Stability Issues

In cases when JIRA's performance deteriorates or if the system becomes unstable or unresponsive, it is important to achieve two goals:

1. Bring system back to normal in the shortest amount of time.
2. Collect information that would help analyze the problem and make sure it does not appear again.

The second goal is strategically very important, however, it might get overlooked in a rush to make things work "now". For example, JIRA administrator may be inclined to restart a stuck JIRA instance quickly in order for it to get back to working state as fast as possible. But if thread dumps are not collected, the developers will never know where JIRA was stuck, so the same problem may happen again.

The first goal is of course also very important. Sometimes JIRA administrator manages to restore system functioning, sometimes help from Atlassian and ALM Works support teams is needed. Support engineers and developers would typically take into account all information they have, analyze it and try to pinpoint the source of the problem. Often additional information is required from the JIRA administrator, and sending requests and replies back and forth takes precious time.

This article is intended to provide JIRA administrators with advice about how to collect maximum information about a performance or stability problem, when that problem happens. The list is not intended to be complete, additional information may still be needed, however, providing all listed information gives a good chance that a support engineer will be able to identify a problem and provide advice sooner.

- Thread Dumps(see page 556)
- Verbose Logging(see page 557)
- Support Zip(see page 557)
- Browser Console Log(see page 557)
- HAR Report(see page 558)
- Screenshots or Video(see page 558)

5.20.7.1 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.

127 mailto:support@almworks.com

- Please refer to Atlassian documentation on generating a thread dump[128] 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.

5.20.7.2 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.

5.20.7.3 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.

5.20.7.4 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.

---

128 https://confluence.atlassian.com/adminjiraserver071/generating-a-thread-dump-802593021.html

Also, please include browser type and version, as well as the information about operating system.

5.20.7.5 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.

5.20.7.6 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.

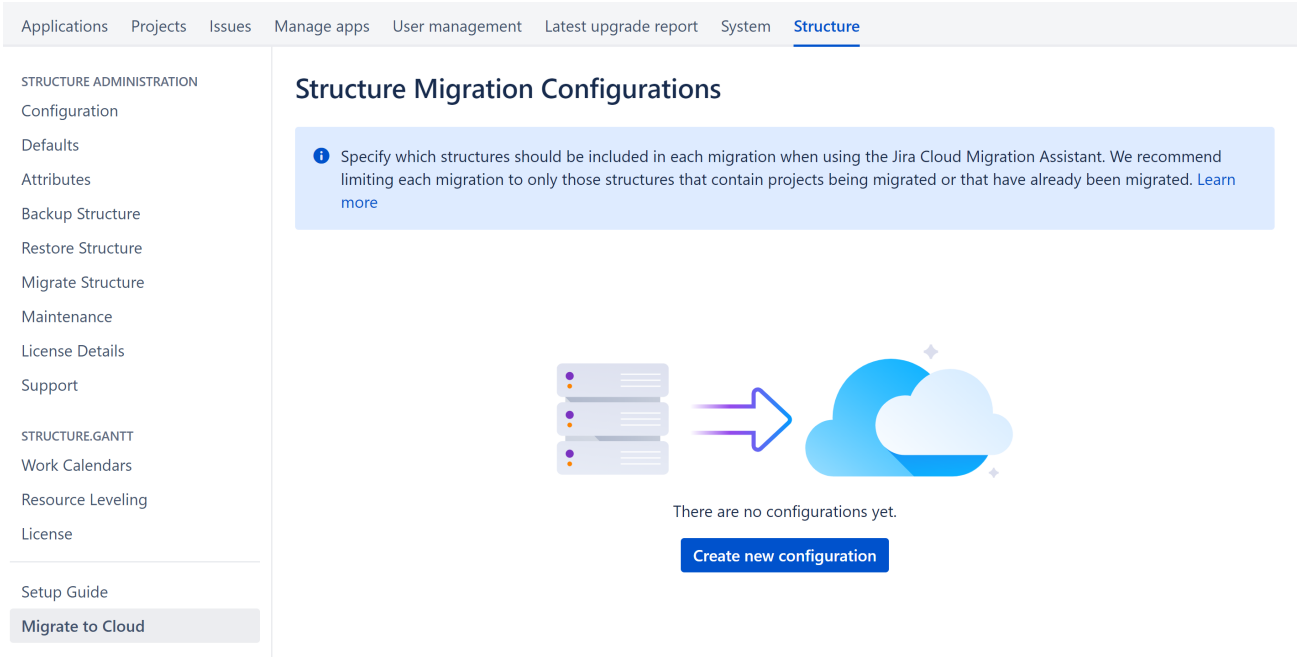
You can use operating system's native tools to capture a video, or install a third-party tool for that. Feel free to ask ALM Works Support for recommendations if you don't have preferable screen capturing tool.

5.21 Migrate to Cloud

The Structure Cloud portion of the migration tool is still under development, and will be released shortly. At this time, it is only possible to prepare migration sets. As soon as the Structure Cloud update is released, any sets you've created can then be used to migrate to Structure Cloud.

Structure integrates with the Jira Cloud Migration Assistant[129] (JCMA) to make it easy to migrate your existing structures to Jira Cloud.

129 https://marketplace.atlassian.com/apps/1222010/jira-cloud-migration-assistant



The Structure Migration Configurations allow you to associate specific structures with JCMA migrations. Then when you run JCMA, those structures will be recreated in Structure Cloud, based on:

- Manually added issues [130] and folders[131]
- Items added and organized through Automation[132]
- Structure views - including the default view(see page 360) and associated views(see page 348) for each structure

Please note that Structure can only add issues from projects that have been migrated to cloud using the JCMA. We highly recommend that you only add structures to a migration configuration if 1) the associated projects are also being migrated in that set, or 2) those projects have been migrated previously using the JCMA.

⚠ At this time, some features are not available in Structure Cloud, including:

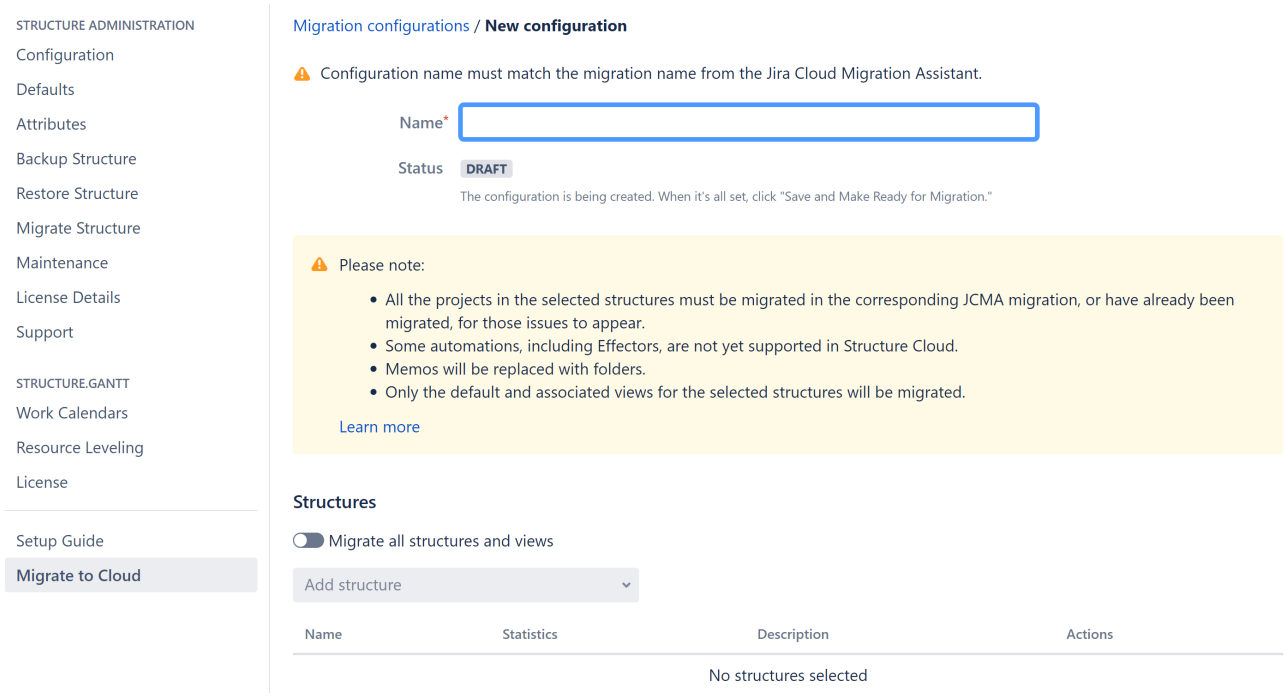
- Certain types of Automation, including Effectors
- Memos – any existing memos will be converted to folders in Structure Cloud

For more information see: Comparison to Structure for Jira Server, Data Center and Service Desk[133].

5.21.1 Creating a Migration Configuration

When you create a new migration configuration, you are telling the JCMA which structures to include with a migration.

130 https://wiki.almworks.com/display/structure2gmaster/Adding+Issues
131 https://wiki.almworks.com/display/structure2gmaster/Folders
132 https://wiki.almworks.com/display/structure2gmaster/Automation
133 https://wiki.almworks.com/display/strcloud/Comparison+to+Structure+for+Jira+Server%2C+Data+Center+and+Service+Desk



To set up a configuration, enter the following:

- **Name:** In order to make it work successfully, it's important that each configuration's name match the corresponding migration.
- **Structures:** Select the structures that should be included in the migration. If you want to include all structures, toggle **Migrate all structures and views**.

We recommend only including structures that have projects also being migrated or that have already been migrated by JCMA. If a project hasn't been migrated using JCMA, any issues from that project will not appear in the structure (even if a project of the same name exists in Jira Cloud).

You can create as many migration configurations as you need, based on how many migrations you're doing with JCMA - so there's no need to add all your structures to a single configuration.

6 Frequently Asked Questions

- Data Center Approved Apps FAQ(see page 561)
  - What are Data Center approved apps?(see page 561)
  - Didn’t Structure already work with Data Center?(see page 562)
  - Why has Structure made this change?(see page 562)
  - What are the criteria for being Data Center approved?(see page 562)
  - Why should I upgrade to the Data Center approved app version?(see page 562)
  - Why is the price different for Data Center approved apps?(see page 563)
  - Do I have to switch to the Data Center approved version? Can I continue to use the server app in Data Center?(see page 563)
- Cannot Create an Issue With +Next Issue (+Sub-Issue) Because of the Required Fields(see page 563)
- Add-on Manager Says Structure Is Unlicensed(see page 564)
- No Check Mark Displayed for a Resolved Issue(see page 564)
- Structure Won't Start(see page 565)
- After an Issue is Moved to Another Project, It Cannot Be Found in the Structure(see page 567)
- User Cannot Access Structure(see page 568)
- Where Do I Find the Jira Server ID?(see page 568)
- Using Subtasks and Structure(see page 569)
- Difference from Sub-tasks(see page 569)
- Performance Considerations(see page 569)
- How to restore the structure using History(see page 570)
- Can a Structure be Exported to Microsoft Word?(see page 570)
- Convert time data in Excel export to Jira format(see page 570)

6.1 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-scare Data Center environments. If you are currently running the Structure for Jira server app on Data Center, you will need to upgrade to the Data Center version in order to continue receiving support.

The following Frequently Asked Questions will help explain these changes, as well as the timelines Atlassian has set for transitioning to Data Center approved apps.

- What are Data Center approved apps?(see page 561)
- Didn’t Structure already work with Data Center?(see page 562)
- Why has Structure made this change?(see page 562)
- What are the criteria for being Data Center approved?(see page 562)
- Why should I upgrade to the Data Center approved app version?(see page 562)
- Why is the price different for Data Center approved apps?(see page 563)
- Do I have to switch to the Data Center approved version? Can I continue to use the server app in Data Center?(see page 563)

6.1.1 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[134].

6.1.2 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.

6.1.3 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 our 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.

6.1.4 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 addition tests focus on the unique needs of Data Center instances, such as caching, database support, clustered environments, and more.

In order to be considered for Data Center approval, developers must:

1. Complete a 100+ question readiness assessment, including an architectural review
2. Complete additional testing to evaluate their apps in an environment that simulates a large-scale Data Center instance

Once these additional steps are completed, Atlassian evaluates the readiness of each app and makes the final decision of whether an app can be designated Data Center approved.

To learn more about the Data Center approved app criteria, see Developing Apps for Atlassian Data Center Products[135].

6.1.5 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.

---

134 https://www.atlassian.com/licensing/data-center-approved-apps
135 https://developer.atlassian.com/platform/marketplace/developing-apps-for-atlassian-data-center-products/

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.

6.1.6 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[136] and Structure.Pages pricing for Data Center[137] on the Atlassian Marketplace. Structure.Testy[138] remains a free extension to Structure on both Server and Data Center platforms. Structure.Gantt[139] is currently free, but will become a paid app starting with version 2.0. You can find more details here(see page 563).

6.1.7 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.

6.2 Cannot Create an Issue With +Next Issue (+Sub-Issue) Because of the Required Fields

6.2.1 Question

I have a number of fields required for new issues. When I try to use Structure's **+Next Issue** or **+Sub-Issue** button, the creation of the issue fails, because the values of the required fields were not provided.

6.2.2 Answer

There are two ways to create new issues when additional fields are required:

6.2.2.1 1. Switch to the Create Issue Dialog

At the bottom of the Add Issue panel, click the Switch to Dialog link to access additional fields. See Creating New Issues: Create Issues Using the "Create Issue" Dialog(see page 112) for more information.

6.2.2.2 2. Add the Required Field to the View

You can enter additional fields when creating a new issue, as long as the field is part of the current view. See Creating New Issues: Editing Other Fields During Creation(see page 112) for more information.

---

136 https://marketplace.atlassian.com/apps/34717/structure-for-jira-projects-at-scale?hosting=datacenter&tab=pricing
137 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=datacenter&tab=pricing
138 https://marketplace.atlassian.com/apps/1212033/structure-testy-test-checklists?hosting=datacenter&tab=overview
139 https://marketplace.atlassian.com/apps/1217809/structure-gantt-planning-at-scale?hosting=datacenter&tab=overview

If the initial creation of an issue has failed, you don't have to lose the entered data. Just add the required fields and double-click on the value you need to edit, or click the **Edit** button in the toolbar. You can change the values of the new issue and try to create it again.

6.3 Add-on Manager Says Structure Is Unlicensed

6.3.1 Question

I have a valid license installed. Why do I see Structure as **Unlicensed** or having **Action Required** in the Add-on Manager?

6.3.2 Answer

That may be so because Add-on Manager is not aware of ALM Works licenses. To verify the true status of your Structure license, check the **Administration | Structure | License Details** page. If it indicates the license is OK, you can safely ignore the status of the Structure license in Add-on Manager.

Structure supports two kinds of licenses — purchased via Atlassian and issued by ALM Works. For details, please see Setting Up Structure License(see page 473).

6.4 No Check Mark Displayed for a Resolved Issue

6.4.1 Question

Why do I see a resolved issue in Structure, but there's no green check mark, which usually indicates that an issue is resolved?

This article answer these questions as well:

- Why do I see a check mark on a unresolved issue?
- Why does an open issue that still in the work have 100% progress indication?
- When I turn on "Unresolved" filter button, why do I see some of the resolved issues anyway?

6.4.2 Answer

The JIRA's notion of a "Resolved Issue" (or "Completed Issue") can be quite confusing. The source of confusion is that an issue is considered to be resolved based on its **Resolution** field, not based on its Status:

- **Unresolved** means that the Resolution field is empty, regardless of issue Status.
- **Resolved** means that the Resolution field has some value, regardless of issue Status.

If an issue has a non-empty Resolution field (i.e. considered Resolved):

- The green check mark is displayed in Structure on that issue;
- The issue is filtered out by the Unresolved button;
- The progress of the issue is 100% regardless of other fields.

See also: Flags Column(see page 317), Filtering(see page 564), Progress Column(see page 300)

6.4.2.1 Problems Caused By Custom Workflows

The default workflow in JIRA contains the "Resolved" status and if you select this status, JIRA requires you to select some non-empty value for the Resolution field too. Thus, the issue gets the Resolved status and becomes truly resolved (or completed), because it has a value in the Resolution field.

The confusion may arise, if in a custom workflow / screen configuration, Resolution field is not set as required or not added to the screens, associated with transitions to the Resolved status. In this case, a user may move an issue to the Resolved status, but the issue will still be unresolved/uncompleted, because the Resolution field is still empty.

If you have such a configuration, in the Structure this problem may manifest itself when you are trying to use the Unresolved filter button (which works as a shorthand for filtering using JQL: "`Resolution is EMPTY`"). The issues with the Resolved status but with no Resolution will still be visible even if you switch the filter on.

Solution:

1. Edit your workflow: in all transitions to a status that should be considered resolved, use a screen with the Resolution field.
2. In all transitions to a status that should not be considered resolved, use "Clear Resolution" step.
3. Make Resolution field required. (It will matter only if Resolution is added to the screen configuration.)
4. Check all screens - "Edit Issue" screen and all screens not mentioned in (1) above should not contain Resolution field.

6.4.2.2 Problems Caused By Manually Added "Unresolved" Resolution Value

To make matters worse, sometimes JIRA administrators add a new resolution option, named "Unresolved". Then, for example, on the workflow's "Reopen" step configuration, instead of clearing the Resolution, they change it to this "Unresolved" value.

The problem is that the new "Unresolved" resolution is still a non-empty value, and any issue having this value in the Resolution field will be considered resolved, by JIRA and Structure and other plugins.

But on the issue page, the user will see *Resolution: Unresolved*. So it will be practically impossible to distinguish this resolved (completed) issue from the issues which are really unresolved (have empty Resolution field).

Solution:

1. Use JIRA's Bulk Change to clear resolution from all issues that have Resolution "Unresolved".
2. Remove resolution "Unresolved".

6.5 Structure Won't Start

6.5.1 Question

I try to install (enable) Structure, but it doesn't work. When I reload the Add-on Manager page, Structure is disabled. What is the problem?

6.5.2 Answer

Structure may fail to start due to the following reasons. To better understand what's going on, check the Jira logs (`catalina.out` or `jira-application.log`) and verify each of the following possible causes.

6.5.2.1 Structure database cannot be created or opened; filesystem is read-only or full

Structure stores all its data in the `structure/` sub-directory of the Jira home directory. At its first launch, it tries to create that directory and shuts down if it fails to do so. At every subsequent start, it tries to open the database contained there and also shuts down if it fails to do so. In all cases, there should be a big warning or error message in the Jira log.

Possible actions:

- Create the `structure` sub-directory manually and grant full permissions on it to the account that is used to run Jira.
- Verify that filesystem is not read-only.
- Verify that there's enough free disk space (at least 100 MB).
- Verify that Structure's database is not opened with some other tool, such as Derby console.

See also: Structure Files Location(see page 498)

6.5.2.2 Some of the required system plugins are disabled

Structure relies on some system plugins. If they are disabled, you may get all kinds of weird messages from Jira when it tries to start Structure. In some cases, the error messages will be completely unrelated to the disabled plugins.

The following is an example of a message you may receive:

```
com.atlassian.plugin.PluginParseException: Unable to load the module's display
conditions: Could not load 'com.almworks.jira.structure.web.UserCanCreateStructureCon
dition' in plugin com.almworks.jira.structure
... stack trace ...
Caused by: com.atlassian.plugin.web.conditions.ConditionLoadingException: Could not
load 'com.almworks.jira.structure.web.UserCanCreateStructureCondition' in plugin
com.almworks.jira.structure
... stack trace ...
Caused by: java.lang.IllegalStateException: Cannot autowire object because the Spring
context is unavailable.  Ensure your OSGi bundle contains the 'Spring-Context'
 header.
... stack trace ...
```

Possible actions:

- Open **Administration | Plugins | Manage Plugins** and click **Show System Plugins**. Verify that all plugins are enabled. If some are disabled, enable them. Then try to enable or reinstall Structure.

If for some reason you need to keep some of the plugins disabled, and Structure won't start without them, please write to support@almworks.com[140].

140 mailto:support@almworks.com

6.5.2.3 Incomplete download or corrupt plugin JAR file

It is possible the Add-on Manager only downloaded a partial JAR file for Structure, if there were any problems with the server or the connection. We have also had reports of Internet Explorer incorrectly downloading the manual install file, turning the JAR file into a ZIP file with invalid content.

Possible actions:

- To verify that you have a correct JAR file, locate the plugin JAR in `plugins/installed-plugins` directory under your Jira home. Structure has the word "structure" in its file name. Verify that the JAR file MD5 hash is the same as listed on the Download Archive(see page 979) page.

6.5.2.4 Incorrect Jira setup

Check the Jira application logs[141] 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[142]. 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[143]. 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[144].

6.6 After an Issue is Moved to Another Project, It Cannot Be Found in the Structure

6.6.1 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?

141 https://confluence.atlassian.com/pages/viewpage.action?pageId=16121981
142 https://confluence.atlassian.com/display/JIRA/Adding+the+Activity+Stream+Gadget
143 https://confluence.atlassian.com/display/JIRA/Updating+JIRA+Plugins+for+JIRA+4.2?focusedCommentId=228623879#comment-228623879
144 mailto:support@almworks.com

6.6.2 Answer

There may be a couple of reasons for this:

6.6.2.1 The New Project Is Not Structure Enabled

Make sure the project where the issue was moved to is enabled for Structure(see page 477). Structure ignores issues in the projects that are not Structure-enabled, so the moved issue is ignored as well, as if it ceased to exist. If you need this issue in the structure, either include the project where the issue resides now into the list of Structure-enabled projects(see page 477) or move the issue to an already Structure-enabled project (such as the original project).

6.6.2.2 The Issue Was Removed Based on Automation Rules

If the structure was built using Automation(see page 122), make sure the issue's new location satisfies the rules of your Insert(see page 125) and/or Extend (see page 146)generators. If it does not, you will need to change your generator rules or move the issue.

6.7 User Cannot Access Structure

6.7.1 Question

The user cannot see the Structure menu or access any structures. How do I resolve this problem?

6.7.2 Answer

To access Structure:

1. Structure has to be enabled for the user(see page 478) - Check which groups the user belongs to and ensure at least one of them has Structure access.
2. Structure must be enabled in any project(see page 477) that the user has permissions to view - Check which projects the user has permissions to view in Jira and make sure they are enabled for Structure.

6.7.2.1 If the user's permissions were just changed:

Configured permissions related to Structure are cached on the server, so for a couple of minutes after permissions are changed the user may not be able to access Structure. These caches will last for approximately 5 minutes before they automatically refresh. After that, the user should be able to use Structure.

To force a cache refresh, the user should do a *hard refresh* of their browser on any Jira page. After that, they should be able to use Structure immediately. In most browsers, hard refresh is achieved by clicking the Refresh button while holding the `Ctrl` or `Shift` button. If that doesn't work, check this list of ways to do a hard refresh in all popular browsers on Wikipedia: http://en.wikipedia.org/wiki/Wikipedia:Bypass_your_cache.

6.8 Where Do I Find the Jira Server ID?

Structure license is tied to a particular Jira Server, and a Server ID is required to generate a Structure license.

The Jira administrator can look up the 16-digit Server ID under **Administration | System Info** or **Administration | Structure | License Details**.

6.9 Using Subtasks and Structure

6.9.1 Question

Should I disable sub-tasks to use Structure?

6.9.2 Answer

Not necessarily. While Structure can be a good replacement for sub-tasks, they can be used in parallel — for example, if you want to try Structure on a single project without affecting other Jira users.

Structure treats sub-tasks as any other issues. If you're using Automation, you can also include a Sub-Tasks Extender(see page 150) to automatically add Jira sub-tasks beneath their parent issues.

6.10 Difference from Sub-tasks

6.10.1 Question

How is issue hierarchy provided by Structure different from the standard sub-tasks?

6.10.2 Answer

Sub-tasks have several major limitations:

- Sub-tasks are only a one-level hierarchy
- Sub-tasks are separate issue types
- Sub-tasks always inherit project and security level from their parent task

None of these limitations are present in Structure. At the same time, Structure provides all the features that sub-tasks have, and more. You can even use existing sub-tasks to build your structure's hierarchy, using the Sub-tasks Extender(see page 150).

6.11 Performance Considerations

If you have a large Jira instance (hundreds of thousands of issues), there are a few things to bear in mind when working with Structure.

- The recommended limit for the number of issues in one structure is 100K. Even at this point, Structure already might be working noticeably slower, especially if there are many users working with the Structure Board at the same time. If you have more issues than that, we recommend that you distribute the issues between several smaller structures (5-10K issues per structure works well). Having a large number of structures has less of an impact on performance.
- Structure's automation feature can place an additional load on servers, particularly as users are first learning to optimize their structures. Structure has some built-in protections, such as pausing

Automation(see page 172) when generation times take too long, but you can also limit the access to Automation by changing permission to access Automation(see page 480).
- If necessary, you can switch off some parts of Structure(see page 498) to reduce the load (for example, the Structure panel on the Issue Page) and limit the group of users Structure is exposed to (see Structure Best Practices(see page 459)).
- If you experience other performance issues, please contact ALM Works Support[145].

6.12 How to restore the structure using History

There may be times when you need to restore a structure to some previous state. You can do so using the History panel.

To restore a structure using History:

1. Delete all current items in the structure before restoring its previous version from the history.
2. Open the structure History(see page 375) panel.
3. In the history, find and select the moment when the structure was in the desired state (before the unwanted changes took place).
4. Press CTRL+A (⌘+A on Macs) to select all issues and press CTRL+C (⌘+C) to cut them to clipboard.
5. Switch off history panel and press CTRL+V (⌘+V) – this should rearrange the structure according to the view you selected in the history.

> If you have some complicated synchronizers(see page 514) (for example, ones which use S-JQL in their configuration), we recommend temporarily disabling the synchronizers before restoring. After the restore is finished, you can enable them and run the resync.

6.13 Can a Structure be Exported to Microsoft Word?

Exporting to Microsoft Word is available through our integration with Xporter[146].

If you do not have Xporter installed, the following built-in export options are available:

- Export to Excel (see page 378)- Export your hierarchies into a fully-functional MS Excel spreadsheet.
- Export to Printable Page(see page 380) - In addition to creating a printer-friendly version of the structure, you can also use turn this into a shareable PDF file using your browser's Print to PDF option.

6.14 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.

145 mailto:support@almworks.com
146 https://www.getxporter.app/blog/xporter-integration-with-structure/

6.14.1 How to Install

1. Download the attached Macro: FormatTimetrackingData.xlam[147].
2. In Excel, open **File** | **Options** | **Add-Ins.**
    a. Select **Manage** | **Excel Add-Ins** and click **Go**.
    b. In the dialog, click the 'Browse' button and select the downloaded `FormatTimetrackingData.xlam`
    c. Make sure the checkbox is selected for the 'FormatTimetrackingData' option and click **OK**.
3. In Excel, go to **File** | **Options** | **Customize Ribbon.**
    a. On the right, select the tab where the button for running the macro will go (e.g., *View).* Select the desired tab.
    b. Click **New Group** to add a custom group in the selected tab.
    c. Select the added group and click **Rename**. Select an icon and a name for it and click **OK**. A button for using the macro will be created.
    d. Go to **Choose commands from** | **Macros**, select FormatTrackingData, click **Add** and **OK**.

6.14.2 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".

147https://wiki.almworks.com/download/attachments/62756887/FormatTimetrackingData.xlam?api=v2&modificationDate=1549373146000&version=1

7 Release Notes

- Structure 6.4 Release Notes(see page 572)
- Structure 6.3 Release Notes(see page 577)
- Structure 6.2 Release Notes(see page 590)
- Structure 6.1 Release Notes(see page 598)
- Structure 6.0 Release Notes(see page 605)
- Structure 5.6 Release Notes(see page 617)
- Structure 5.5 Release Notes(see page 626)
- Structure 5.4 Release Notes(see page 629)
- Structure 5.3 Release Notes(see page 635)
- Structure 5.2 Release Notes(see page 641)
- Structure 5.1 Release Notes(see page 648)
- Structure 5.0 Release Notes(see page 655)
- Structure 4.6 Release Notes(see page 660)
- Structure 4.5 Release Notes(see page 669)
- Structure 4.4 Release Notes(see page 675)
- Structure 4.3 Release Notes(see page 678)
- Structure 4.2 Release Notes(see page 684)
- Structure 4.1 Release Notes(see page 690)
- Structure 4.0 Release Notes(see page 698)
- Structure 3.6 Release Notes(see page 704)
- Structure 3.5 Release Notes(see page 708)
- Structure 3.4 Release Notes(see page 716)
- Structure 3.3 Release Notes(see page 723)
- Structure 3.2 Release Notes(see page 732)
- Structure 3.1 Release Notes(see page 739)
- Structure 3.0 Release Notes(see page 742)
- (HIDDEN) Release Notes (Structure 1-2)(see page 761)

7.1 Structure 6.4 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[148]

7th of December, 2020

Structure 6.4 adds Fix Version and Sprint data in the Summary column and formulas, performance improvements and more

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[149] (no installation required)

148 http://alm.works/covid19
149 https://demo-structure.almworks.com/

7.1.1 Version Highlights

Structure 6.4 adds:

- Fix version and sprint details can now be shown in the Summary column and used in formulas
- Option to show names, avatars or both in user columns
- Additional improvements and fixes

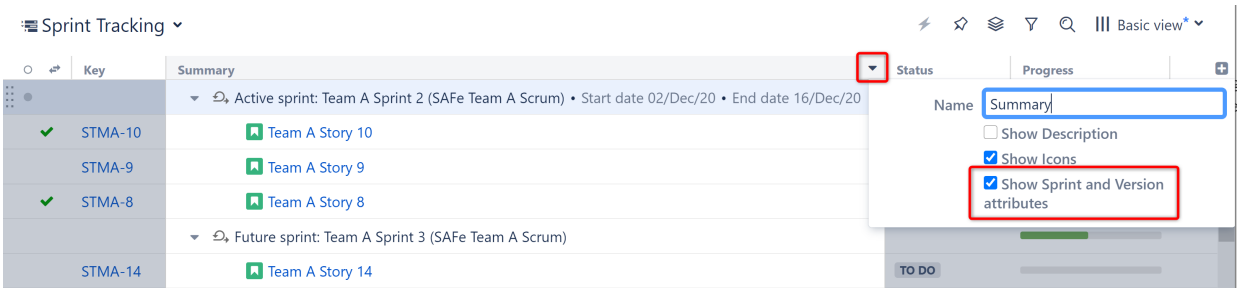
7.1.2 Changes in Detail

7.1.2.1 Fix Version and Sprint data

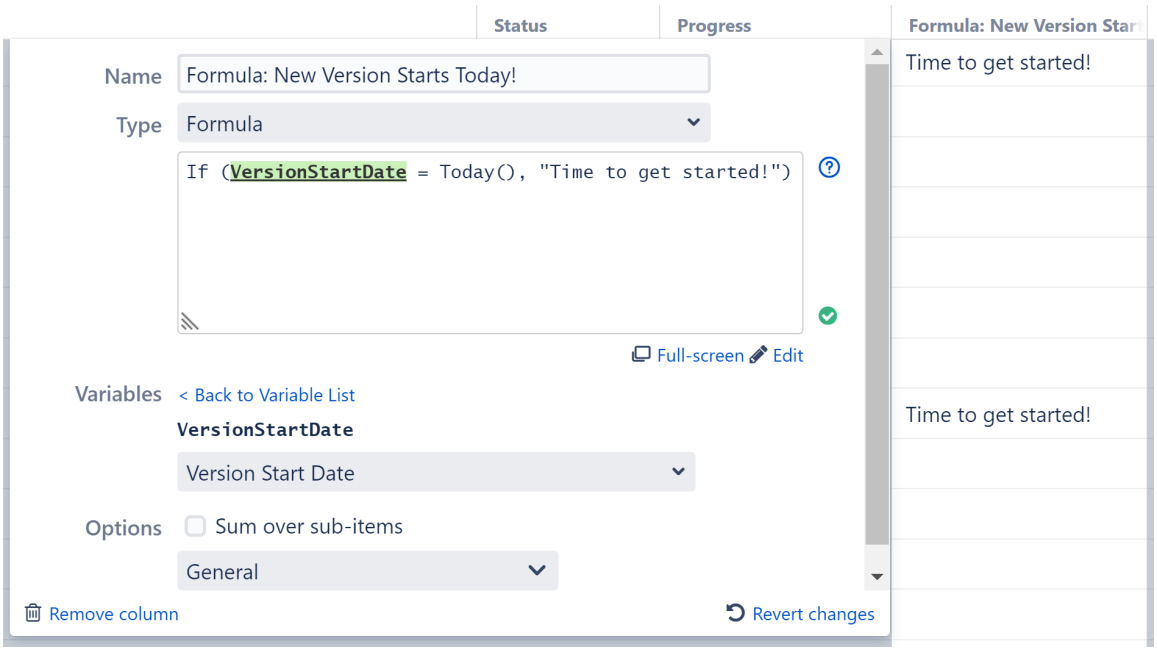
The following attributes can now be shown in the Summary column, or used in formulas and effectors:

- Fix version start date
- Fix version release date
- Fix version description
- Sprint start date
- Sprint end date
- Sprint Goal

To show these attributes in the summary column, click the arrow icon in the column header and check **Show Sprint and Version attributes**.

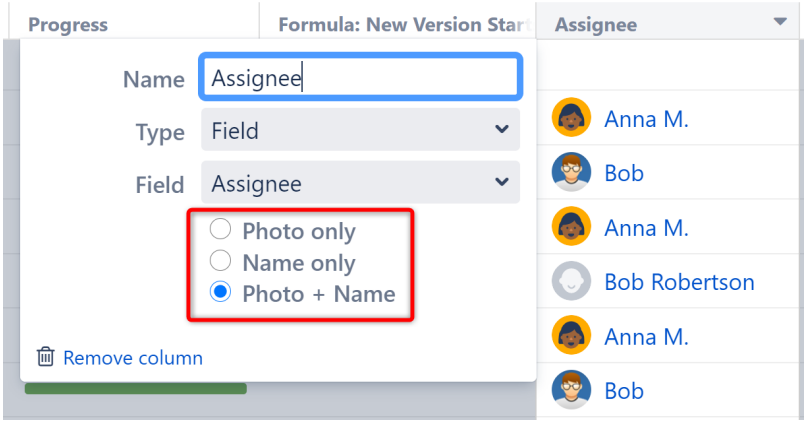


To include these in a formula, map a variable to the appropriate version or sprint attribute.



7.1.2.2 Photo and Name in User Fields

It is now possible to choose whether user fields (Assignee, Creator, Reporter, custom user-picker fields) show the user's name, photo (avatar) or both.



7.1.2.3 Additional Updates

- It is now possible to select multiple projects and version when using the Backlog Structure wizard.
- Extender performance improvements. Learn more(see page 576)
- Synchronizers now check the structure size before creating new rows. Learn more(see page 577)
- Fixed: When working on the Project Page, Structure did not auto-switch to the default.
- Fixed: When viewing a structure in two panels, changing an issue's attributes in one panel would cause attributes in the other to be marked as outdated (grayed out).

7.1.3 Supported Versions

Structure 6.4 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Note, that 6.4 is the last version that supports Jira 7.13; the next major version will require Jira 8.5+.

Compatible plugins versions:

- Structure.Gantt[150] 2.7+
- Structure.Pages[151] 1.5+
- Structure.Testy [152] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.1.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.1.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [153]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.1.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.3 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

150 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
151 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
152 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
153 https://wiki.almworks.com/display/structure2gmaster/Download

7.1.4.3 Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.4, either from the Atlassian Marketplace or our Download[154] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 49).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.1.5 Enterprise Deployment Notes

Structure 6.4 introduces a few changes and improvements especially important for large-scale Jira Server and Jira Data Center instances.

7.1.5.1 Extender Performance Improvements

We have updated our structure generation algorithm, which now allows extenders to preload the data for all rows on the current level of the structure at once, even when those rows belong to different sub-trees. Combined with new ways of loading data for multiple issues at once, the new algorithm has led to significant performance improvements for extender-based structures.

All four extenders that ship with Structure have been updated to take advantage of the new generation algorithm, and in our internal tests we have seen them work several times faster in certain scenarios. If you have large, multi-level structures built with extenders, we advise that you verify their performance on a staging system before upgrading.

7.1.5.2 Confluence Gadget Updates

Continuing the work started in Structure 6.3, we have made a few adjustments in the Structure gadget when it's embedded in Confluence pages. On Jira Data Center, we have increased the gadget's poll interval to 30 seconds and also increased the window size for loading attribute values. With fewer REST requests from the gadget to the server, the probability of reaching a different cluster node should now be reduced. Also, we have made various aspects of the gadget's communication with the server configurable with dark features.

7.1.5.3 Other Improvements and Fixes

We have identified and fixed a bug where a simple sorter by an issue field would in certain cases cause a full re-generation of the structure on each request, making it constantly busy and, in certain cases, unusable.

---

154 https://wiki.almworks.com/display/structure2gmaster/Download

Another problem we have fixed in this version was related to synchronizers. Since Structure 4.1, there is a hard limit on the size of static structures (100,000 rows by default). Synchronizers are also not allowed to exceed the limit, but we have discovered that they would still insert records for new rows into the database, even when it wouldn't be possible to add those rows to the structure. Since synchronizers run quietly in the background, this could cause a lot of garbage to accumulate in the database. We have fixed the problem, but if you have synchronizers operating on structures that have reached the limit, we advise you to check the size of the `AO_8BAD1B_ROW` table in your database.

7.1.5.4 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 6.4. We advise running the same testing procedures as you've done for previous upgrades.

> Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[155].

7.1.6 API Changes in Structure 6.4

7.1.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.6.0

See Structure API Versions(see page 883) for full version information and downloads.

7.1.6.2 Compatible Changes in the Java API

We have added two new constants to support the "creator" attribute:

- `CoreAttributeSpecs.CREATOR` is the attribute spec that returns the creator of an item as an `ApplicationUser`;
- `CoreAttributeSpecs.Id.CREATOR` is the attribute ID for the "creator" attribute.

7.2 Structure 6.3 Release Notes

> ALM Works Business Continuity During the COVID-19 Outbreak[156]

155 mailto:support@almworks.com
156 http://alm.works/covid19

8th of October, 2020

Structure 6.3 adds the Status Rollup Effector, default views, Project Category grouper and more.

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[157] (no installation required)

7.2.1 Version Highlights

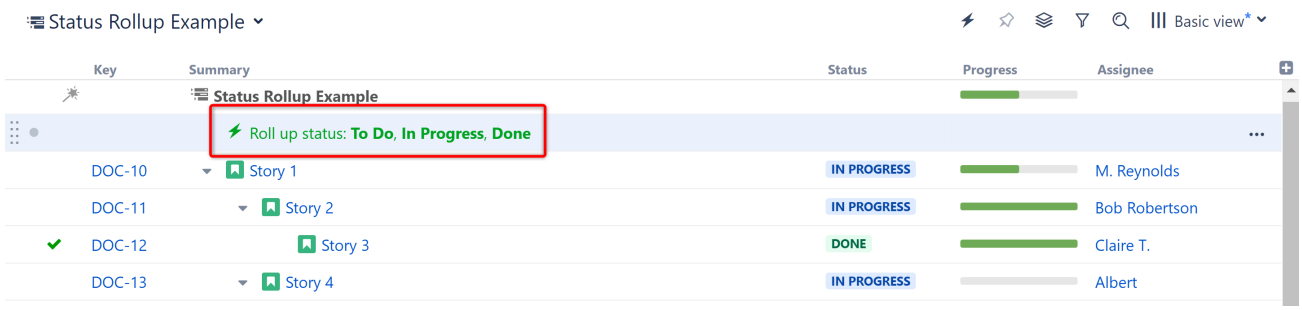
Structure 6.3 adds:

- Status Rollup Effector
- Default view for structures
- Group by Project Category
- Improved Information Center
- Ability to edit numeric custom fields through Structure Gadget
- Additional improvements

7.2.2 Changes in Detail

7.2.2.1 Status Rollup Effector

This new effector allows you to automatically update parent issue statuses based on the earliest status of their sub-issues.

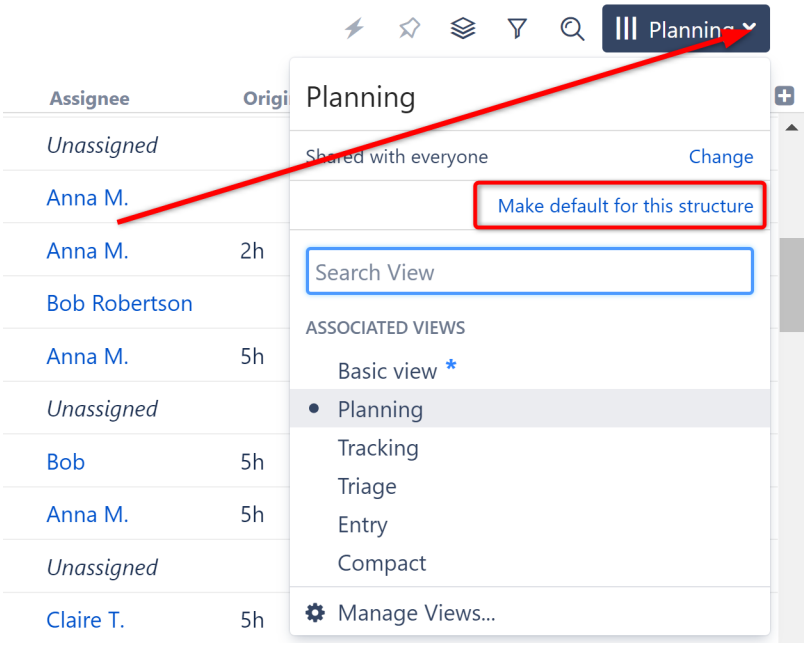


Documentation: Status Rollup Effector(see page 157)

7.2.2.2 Default Views

It is now possible to select a default view for each structure right from the Views menu. When set, users who open the structure for the first time will see the default view. Additionally, if there is no recently-selected view available for the structure, it will open with the default view.

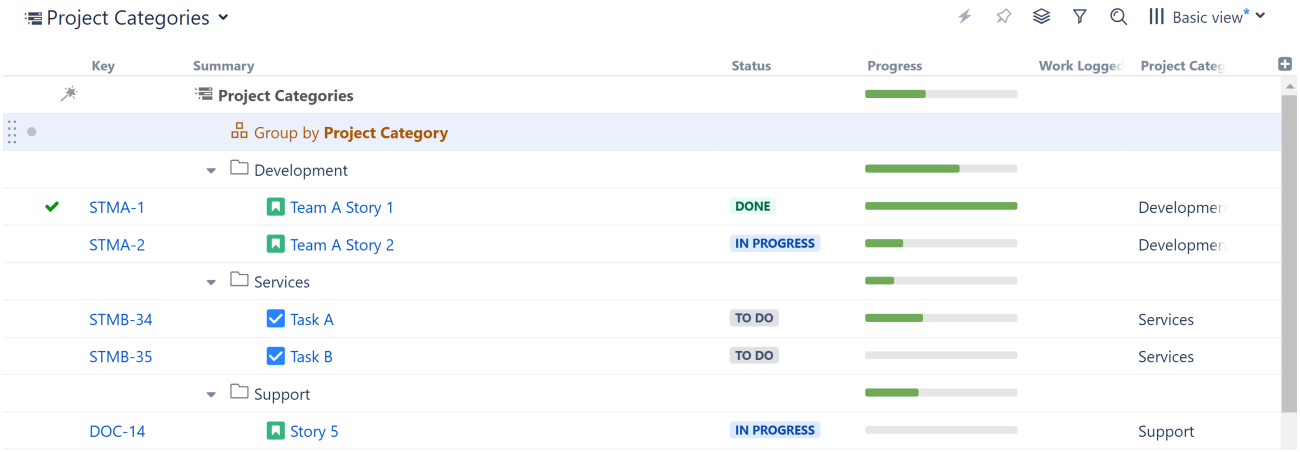
157 https://demo-structure.almworks.com/



Documentation: Views Menu(see page 337)

7.2.2.3 Group by Project Category and ScriptRunner Single Issue Picker

It is now possible to group items by Project Category and the ScriptRunner Single Issue Picker field, using Group Automation.



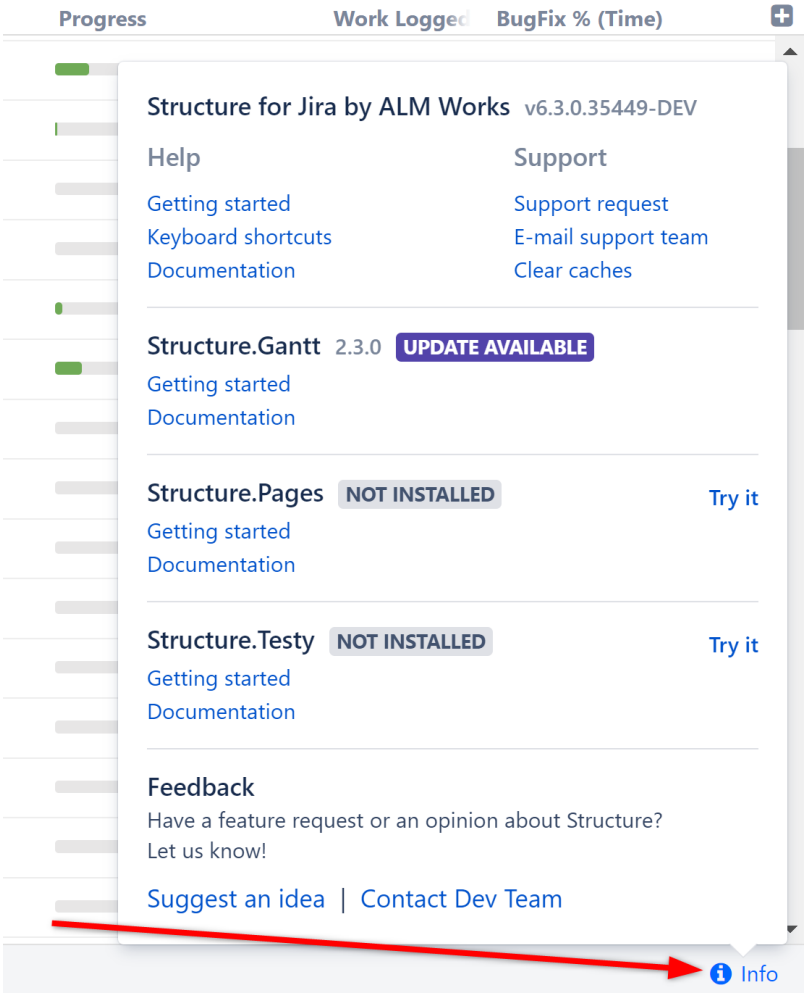
To Group by Project Category, select **Automation | Group by Text Attribute**, and then select **Group by Project Category**. If you have ScriptRunner installed and have defined Single Issue Pickers, they will appear under the **Automation | Group by...** menu.

- Project Category and the ScriptRunner Picker field can also be added as Structure columns and used in formulas.
- ScriptRunner Picker values can be updated within the structure by dragging an issue to another group or using inline editing (when added as a column).

Documentation: Group Generators(see page 141), ScriptRunner(see page 405)

7.2.2.4 Improved Information Center

The improved Info window now includes additional links and information about Structure extensions.



7.2.2.5 Additional Updates

- Structure now remembers the last view used with each structure, and returns to that view the next time the structure is opened.
- A new hotkey ('vvv') returns to the previous view.
- It is now impossible to edit gadget in Confluence.
- It is now possible to update numeric fields in the Dashboard gadget.
- It is now possible to include the backlog when using the Sprint Filter Transformation.
- Query Match is now a resizable field in Structure formulas.
- Attribute system now reads the values of numeric, date, and date-time custom fields from the Lucene index. Learn more.(see page 582)
- Fixed: Data was lost when saving changes to a Memo.
- Fixed: Confluence gadget displayed values unrelated to the displayed structure. Learn more.(see page 582)
- Fixed: Export failed due to Lock Timeout on busy systems. Learn more.(see page 583)

7.2.3 Supported Versions

Structure 6.3 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Desk).

Compatible plugins versions:

- Structure.Gantt[158] 2.4+
- Structure.Pages[159] 1.5+
- Structure.Testy [160] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.2.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.2.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [161]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.2.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

158 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
159 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
160 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
161 https://wiki.almworks.com/display/structure2gmaster/Download

7.2.4.3 Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[162] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 49).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.2.5 Enterprise Deployment Notes

Structure 6.3 introduces a few changes and improvements especially important for large-scale Jira Server and Jira Data Center instances.

7.2.5.1 Reading Custom Fields from the Index

Starting with Structure 6.3, the attribute system reads the values of numeric, date, and date-time custom fields from the Lucene index instead of the database. Reading a single field for many issues from the index is usually much faster than loading a lot of issue objects one by one from the database. Structure generators have been using this technique for a long time, and now we're bringing it to the attribute subsystem.

This improvement is especially important for aggregate values like sums, where Structure sometimes needs to load the values for many more issues than the user sees on the screen to calculate the sum. If you have big structures with thousands of issues and you calculate aggregates based on numeric, date and date-time custom fields on those structures, we advise that you verify the performance of those aggregate columns in a staging environment before upgrading.

7.2.5.2 Cluster Node Tracking in Structure Gadget in Confluence

The Structure gadget, unlike most other Jira gadgets, is a live web application, and it uses multiple REST requests to load the data and keep it up-to-date. When a Structure gadget is inside a Confluence page, those REST requests have to be proxied by Confluence: the gadget calls Confluence, which then calls Jira and forwards its reply back to the gadget. For each forwarded request, Confluence authenticates the current user, creating a new, single-use user session. In a Jira Data Center instance, the load balancer can route these requests to different cluster nodes, so it is possible, for example, that the gadget loads the structure from one node, but the values shown in the grid come from a different node.

---

162 https://wiki.almworks.com/display/structure2gmaster/Download

Structure was not designed to operate in these conditions. This problem was hard to notice in earlier versions of Structure, but since Structure 6.0, completely unrelated values may be shown in the grid, giving the illusion that a different structure is displayed.

In Structure 6.3, we've updated the gadget to keep track of the cluster node: it remembers the node from which it received its first reply, and if later a reply comes from a different node, the gadget will ignore it and retry the request, hoping that the load balancer will redirect it to the original node. If the gadget is unable to reach the original node after multiple attempts, it will stop updating the data and display an error message. If the original node is down, reloading the page will reinitialize the gadget and "bind" it to a different node.

Also, the gadget will always switch to read-only mode if it's on a Confluence page and the Jira instance is a Data Center instance. This solution is only a temporary workaround. We are still researching better ways of integrating Structure into Confluence pages.

7.2.5.3 More Reliable Export Under Load

We have seen a few cases where, on a busy system, exporting a structure to an Excel file or printable HTML would fail due to a lock timeout because the exporting thread would not be able to lock and refresh the structure for 30 seconds. Users would then have to retry or, in the most extreme cases, schedule their exports during a quiet period.

In Structure 6.3, we've made the export process more robust: it will now wait for at least 10 minutes to let the other threads finish and get access to the structure. Also, after 2 minutes of waiting time, the exporter will try to switch to a "fast path": if there's already a recent version of this structure in the cache, it will be exported without waiting for the lock (accompanied by a warning that the exported version may be outdated). We hope that these changes improve users' experience when exporting large, complex structures under heavy load.

7.2.5.4 Testing on Staging Environment

Apart from the changes and suggestions above, there are no particular special areas of interest for load testing and stress testing Structure 6.3. We advise running the same testing procedures as you've done for previous upgrades.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[163].

7.2.6 API Changes in Structure 6.3

7.2.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.4.0

163 mailto:support@almworks.com

See Structure API Versions(see page 883) for full version information and downloads.

7.2.6.2 Compatible Changes in the Java API

The `MapObject` class has received new versions of `getLong()`, `getInt()`, and `getBoolean()` that accept a default value. These are added for convenience.

`StructureViewManager.makeDefaultForStructure()` is a new method that will associate a view with a structure and make it the default view for the given page, all in a single call.

7.2.7 API Changes in Structure 6.3.2

7.2.7.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.5.0

See Structure API Versions(see page 883) for full version information and downloads.

7.2.7.2 Compatible Changes in the Java API

The `StructurePluginHelper` interface has two new methods: `isSystemAdmin()` and `isSystemAdmin(ApplicationUser)` check whether a user is a Jira system administrator. They complement the existing `isAdmin()` methods.

The `BackupOperation` interface has several new methods that allow you to do partial backups and back up to `OutputStream` instead of a file on the server:

- `setBackupStructureIds(LongIterable)`
- `setBackupViewIds(LongIterable)`
- `setBackupAppConfiguration(boolean)`
- `setBackupPerspectives(boolean)`
- `setBackupFavorites(boolean)`
- `backupToStream(OutputStream)`

Three new methods have been added to the `CoreIdentities` class to support memos and improve support for folders:

- `memo(long)`
- `isMemo(ItemIdentity)`
- `isFolder(ItemIdentity)`

7.2.8 Structure 6.3.1 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[164]

10th of October, 2020

Structure 6.3.1 is a patch release for 6.3. It fixes an issue with mentioning multiple users on the Jira issue page.

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[165] (no installation required)

7.2.8.1 Patch Release

Structure 6.3.1 is a patch release for Structure 6.3. It fixes an issue that prevented some users from including multiple user mentions within comments on the Jira issue page.

Upgrade from Structure 6.3 is highly recommended.

7.2.8.2 Supported Versions

Structure 6.3 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Desk).

Compatible plugins versions:

- Structure.Gantt[166] 2.4+
- Structure.Pages[167] 1.5+
- Structure.Testy [168] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.2.8.3 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

164 http://alm.works/covid19
165 https://demo-structure.almworks.com/
166 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
167 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
168 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [169]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[170] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 49).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.2.9 Structure 6.3.2 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[171]

169 https://wiki.almworks.com/display/structure2gmaster/Download
170 https://wiki.almworks.com/display/structure2gmaster/Download
171 http://alm.works/covid19

18th of November, 2020

Structure 6.3.2 is a patch release for 6.3. It adds Jira Cloud Migration Assistant support and fixes a problem with custom fields.

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[172] (no installation required)

7.2.9.1 Patch Release

Structure 6.3.2 is a patch release for Structure 6.3. It provides:

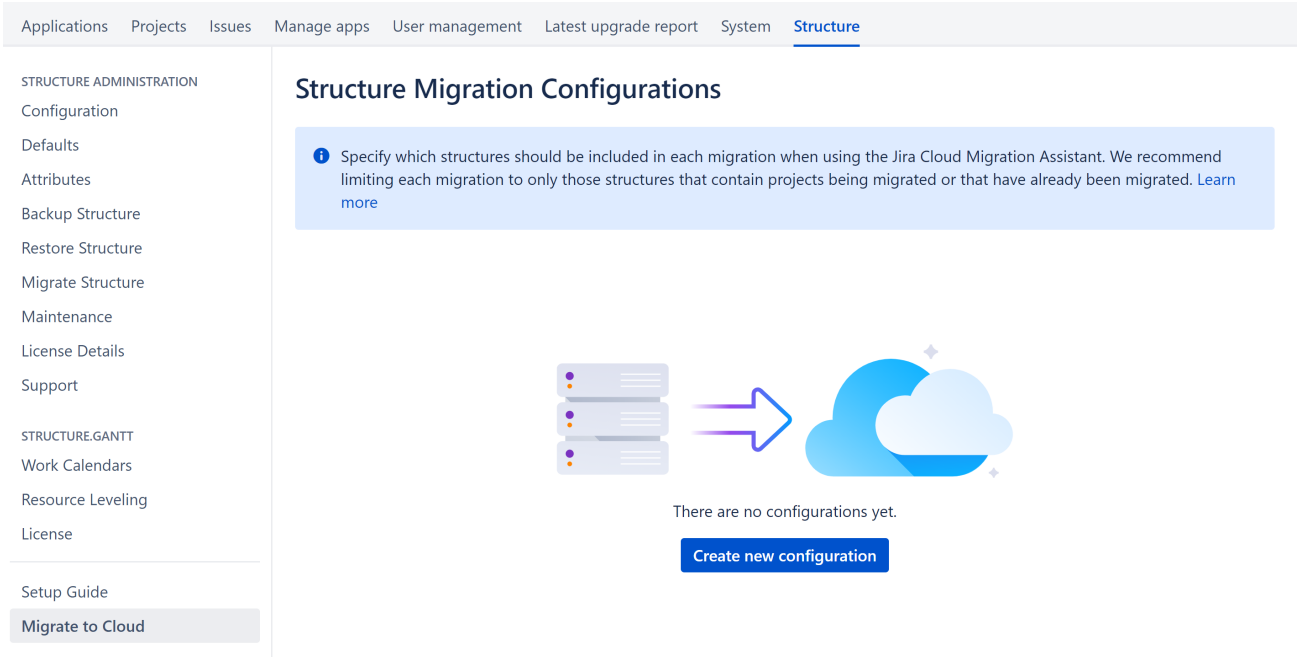
- Migration assistance to migrate structures from Server/Data Center to Jira Cloud
- Fix to an issue that prevented Structure from reading the values of some custom fields

Upgrade from Structure 6.3 is highly recommended.

7.2.9.2 Changes in Detail

Migration Assistance

It is now possible to add Structure data to a migration when using the Jira Cloud Migration Assistant (JCMA).



Structure Migration Configurations allow you to associate specific structures with JCMA migrations. When you run JCMA, those structures will be recreated in Structure Cloud, based on:

- Manually added issues (see page 94)and folders(see page 95)

172 https://demo-structure.almworks.com/

- Items added and organized through Automation(see page 122)
- The default view(see page 360) and associated views(see page 348) for each structure

Please note the following restrictions:

- Structure can only add issues from projects that have been migrated to cloud using the JCMA.
- At this time, some features are not available in Structure Cloud, including:
  - Certain types of Automation, including Effectors
  - Memos – any existing memos will be converted to folders in Structure Cloud

  For more information see: Comparison to Structure for Jira Server, Data Center and Service Management(see page 586).

> ⚠ The Structure Cloud portion of the migration tool is still under development, and will be released shortly. At this time, it is only possible to prepare migration sets. As soon as the Structure Cloud update is released, any sets you've created can then be used to migrate to Structure Cloud.

Documentation: Migrate to Cloud(see page 558)

Additional Updates

- Fixed: Structure was unable to read the values of some custom fields. See Enterprise Deployment Notes(see page 589).

7.2.9.3 Supported Versions

Structure 6.3 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Management/Service Desk).

Compatible plugins versions:

- Structure.Gantt[173] 2.4+
- Structure.Pages[174] 1.5+
- Structure.Testy [175] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.2.9.4 Installation and Upgrade

> ⚠ **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

---

173 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
174 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
175 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [176]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.3.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[177] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 49).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.2.9.5 Enterprise Deployment Notes

Structure 6.3 introduced an optimization in the attribute system, which used the Lucene index to quickly read the values of numeric, date, and date-time custom fields. We have since seen a few support cases where the optimized code didn't work as expected, which resulted in Structure being unable to read the values of certain custom fields. We are working on resolving the problem, but in the meantime Structure 6.3.2 disables this optimization by default, and reads all custom field values from the database. If needed, the optimization can be enabled by setting a dark features(see page 499).

Otherwise, there are no particular special areas of interest for load testing and stress testing Structure 6.3.2. We advise running the same testing procedures as you've done for previous upgrades.

176 https://wiki.almworks.com/display/structure2gmaster/Download
177 https://wiki.almworks.com/display/structure2gmaster/Download

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[178].

7.3 Structure 6.2 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[179]

10th of August, 2020

Structure 6.2 adds Effectors support for duration and multi-value fields, full-screen formula editor, migration improvements, last comment column and more

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[180] (no installation required)

7.3.1 Version Highlights

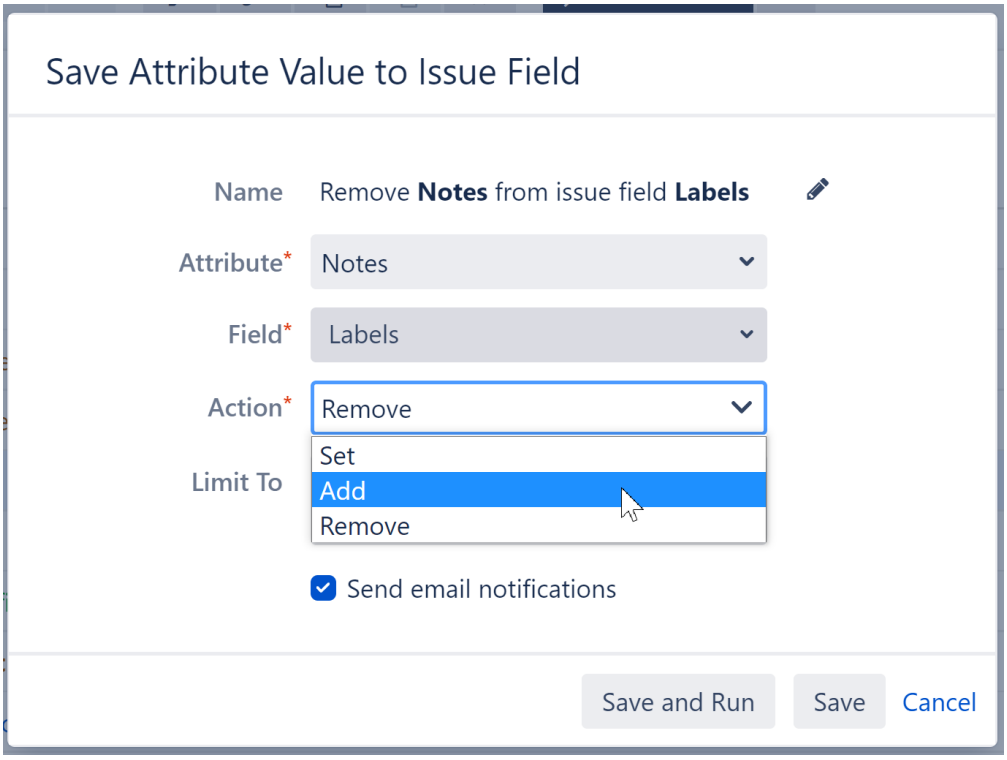
- Effectors can write values to duration and multi-value fields
- Full-screen formula editor
- Last Comment column

7.3.2 Changes in Detail

7.3.2.1 Effectors Can Write Values to Duration and Multi-Value Fields

Effectors can now write values to duration fields and multi-value fields. With multi-value fields, effectors can replace, add or remove values based on a selected attribute.

178 mailto:support@almworks.com
179 http://alm.works/covid19
180 https://demo-structure.almworks.com/



Documentation: Attribute to Issue Field Effector(see page 154)

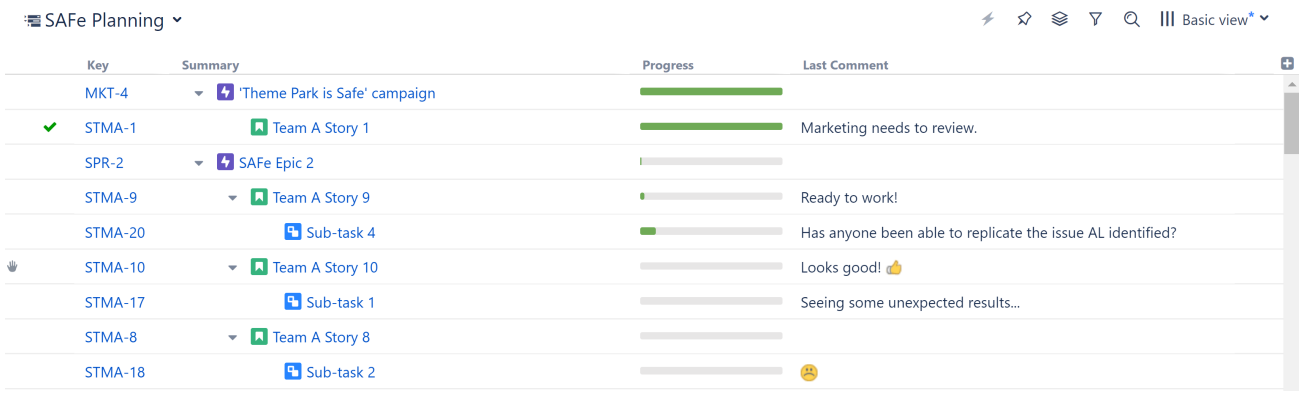
7.3.2.2 Full-Screen Formula Editor

The formula editor can be expanded to more easily write/review a formula.

Documentation: Formula Columns(see page 197)

7.3.2.3 Last Comment Column

The Last Comment column displays the last comment left for each issue.



Documentation: Last Comment Column(see page 330)

7.3.2.4 Additional Updates

- Sort by Status (Automation and Transformation) now sorts by Status Category by default.
- On the Project Page, Structure now opens in the Single Grid layout by default.
- View modifications made with Grid + Details layout are now preserved when changing to Single Grid layout (on Structure Board and Project Page).
- When a client opens a structure for the first time, Structure now checks for recent cached versions on the server to reduce load times - see Enterprise Deployment Notes(see page 594) for details.
- Improved migration - see Enterprise Deployment Notes(see page 594) for details.
- Ability to enable users to manage permissions for groups they are not members of - see Advanced Configuration and Dark Features(see page 499)
- Fixed: Data was lost when saving changes to a Memo

7.3.3 Supported Versions

Structure 6.2 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Desk).

Compatible plugins versions:

- Structure.Gantt[181] 2.4+
- Structure.Pages[182] 1.5+
- Structure.Testy [183] 2.4+

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

181 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
182 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
183 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.3.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.3.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [184]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.3.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.3.4.3 Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[185] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 49).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.

---

184 https://wiki.almworks.com/display/structure2gmaster/Download
185 https://wiki.almworks.com/display/structure2gmaster/Download

5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.3.5 Enterprise Deployment Notes

Structure 6.2 introduces several improvements especially important for large-scale Jira Server and Jira Data Center instances.

7.3.5.1 Faster Initial Loading of Structures from the Cache

One of the changes in Structure 6.2 aims to improve responsiveness in certain scenarios. Previously, when a user opened a structure for the first time, the client would send a regular update request to the server to check if anything changed there since the last time this structure was generated and regenerate it as necessary. On large, busy systems and with complex structures that take a long time to generate, this would sometimes cause a significant delay before the user could see the structure.

Starting with Structure 6.2, when the client loads a structure from the server for the first time, it will send a special lightweight request that will first check if the server already has a recent version of this structure in its cache. If such a version exists, the server would quickly return it to the client, without checking for changes or waiting for ongoing regeneration to complete. After receiving and displaying this cached version, the client will then send a regular update request to load the most recent version of the structure in the background. This way, popular and complex structures should now load faster, especially on busy Jira instances with a high rate of change.

7.3.5.2 Migration Improvements

Some of our larger customers have used Migrate Structure to consolidate Structure data from several Jira instances that they were migrating into a single Jira Data Center instance. During the process, they encountered several problems that required manual intervention or writing ad-hoc scripts:

- Owners of Structure views were not migrated.
- Permission rules based on project roles were not migrated.
- "Apply Permissions" rules for structures were not migrated.

To address this, we've made the following improvements in Structure 6.2:

- View owners are now migrated.
- Role-based permission rules are now migrated.
- "Apply Permissions" rules are migrated if the structures they refer to are also migrated.
- Structure and view owners, as well as users mentioned in structure permission rules, are resolved by username first, to better deal with synthetic user keys in the latest Jira versions.
- When a permission rule cannot be migrated, Structure logs a warning and skips only this rule. Previously, all permission rules were skipped when any of them couldn't be migrated.

Please note that if you want all of these improvements, you need to migrate from backup files made with Structure 6.2, as they contain all the necessary information, such as usernames and project role names. Those improvements that don't need new information are also available when migrating from backup files made with earlier versions of Structure.

7.3.5.3 Testing on Staging Environment

To check faster initial loading of structures from the cache, you can try the following:

- Create a complex structure that takes a significant time to generate and update, for example, using a JQL inserter and several extenders. If you already have such a structure, open it and wait for it to load, so that it's generated and stored in the cache.
- Simulate a high rate of change, e.g. by making random changes to issues in that structure using a script.
- Try opening this structure in multiple browser tabs, using different user accounts. Notice the time it takes the structure to appear on the screen.

Apart from this change in the client-server communication, there are no particular special areas of interest for load testing and stress testing Structure 6.2. We advise running the same testing procedures as you've done for previous upgrades.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[186].

7.3.6 API Changes in Structure 6.2

7.3.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.3.0

See Structure API Versions(see page 883) for full version information and downloads.

7.3.6.2 Compatible Changes in the Java API

`CoreEffects` has many new methods that generate effect descriptions for the new effect providers in Structure 6.2, which can update multi-valued fields, both built-in and custom. There are 30 new methods in total, please refer to the Javadoc[187] for details.

There are two new constants in `CoreEffectorParameters`:

- `SEND_NOTIFICATIONS` controls whether e-mail notifications are sent when effects are applied.
- `COLLECTION_OPERATION` controls whether an effector should add, remove, or replace values when working with a multi-valued field.

The Attribute to Issue Field effector supports both parameters.

`StructureConfiguration` has new setter methods that accept project IDs and permission subjects where previously it accepted only string-encoded representations thereof:

- `setPickedProjectIds(LongIterable)`
- `setPickedProjectIds(Collection<Long>)`

186 mailto:support@almworks.com
187 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/CoreEffects.html

- `setPermissionSubjects(StructureAppPermission, Collection<? extends PermissionSubject>)`

`StructurePage.STRUCTURE_BOARD_WITH_DETAILS` is no longer supported and has been marked `@Deprecated`.

New methods `StructureUtil.ivn(String)` and `TypeUtils.ivn(String)` convert a string to a nullable `Integer`. They complement `iv()`, `lv()`, and `lvn()` that these classes already had.

7.3.6.3 New REST API for App Configuration

There is a new REST API for working with the app configuration. It lets you read and update the list of projects for which Structure is enabled, as well as Structure's global permissions. Please refer to the documentation(see page 933) for a detailed description.

7.3.7 Structure 6.2.1 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[188]

2nd of September, 2020

Structure 6.2.1 is a patch release for 6.2. It adds Jira 8.12 compatibility and a security fix.

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[189] (no installation required)

7.3.7.1 Patch Release

Structure 6.2.1 is a patch release for Structure 6.2. It adds Jira 8.12 compatibility and a security fix

7.3.7.2 Supported Versions

Structure 6.2.1 and all extensions support Jira versions 7.13 or later. This release is for customers using Jira Server or Data Center (Jira Core, Jira Software, or Jira Service Desk).

Compatible plugins versions:

- Structure.Gantt[190] 2.4+
- Structure.Pages[191] 1.5+
- Structure.Testy [192] 2.4+

188 http://alm.works/covid19
189 https://demo-structure.almworks.com/
190 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
191 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
192 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

Cloud customers can learn more about our products on the "Cloud" tab of our marketplace listing.

7.3.7.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [193]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[194] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Getting Started with Structure(see page 49).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

193 https://wiki.almworks.com/display/structure2gmaster/Download
194 https://wiki.almworks.com/display/structure2gmaster/Download

7.4 Structure 6.1 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[195]

3rd of June, 2020

Structure 6.1 adds rich text to Memos, the ability to use JQL/S-JQL and text filters in formula, support for Jira Anonymization and more

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[196] (no installation required)

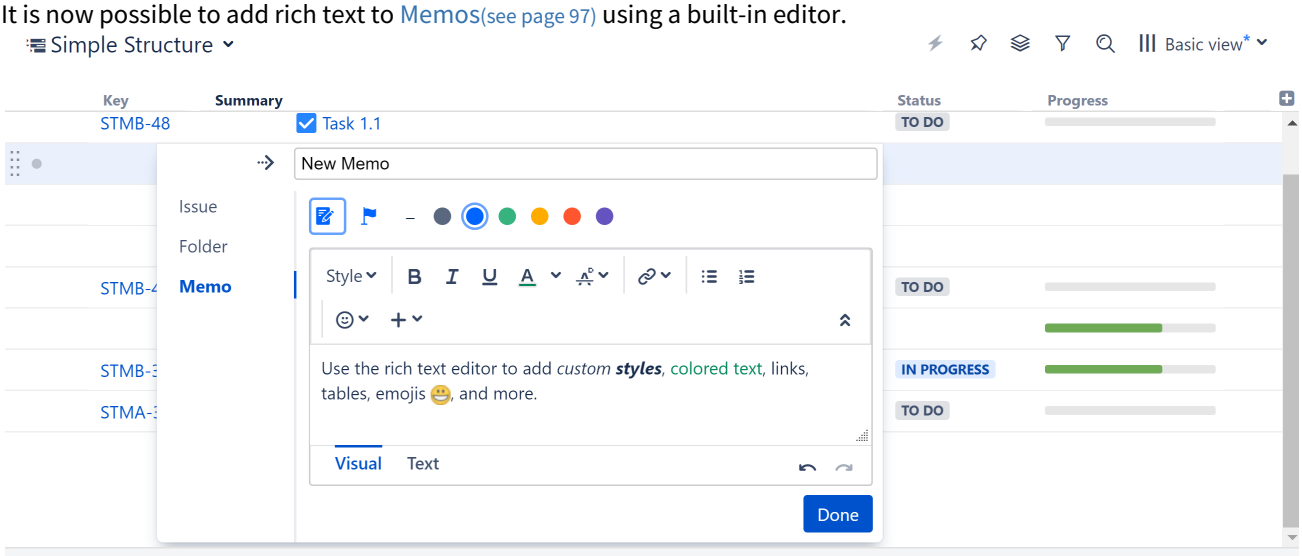
7.4.1 Version Highlights

- Rich text editor for Memos
- JQL, S-JQL and Text Query Match columns
- JQL, S-JQL and Text filters in formulas
- Support for Jira Anonymization

7.4.2 Changes in Detail

7.4.2.1 Rich Text Editor for Memos

It is now possible to add rich text to Memos(see page 97) using a built-in editor.



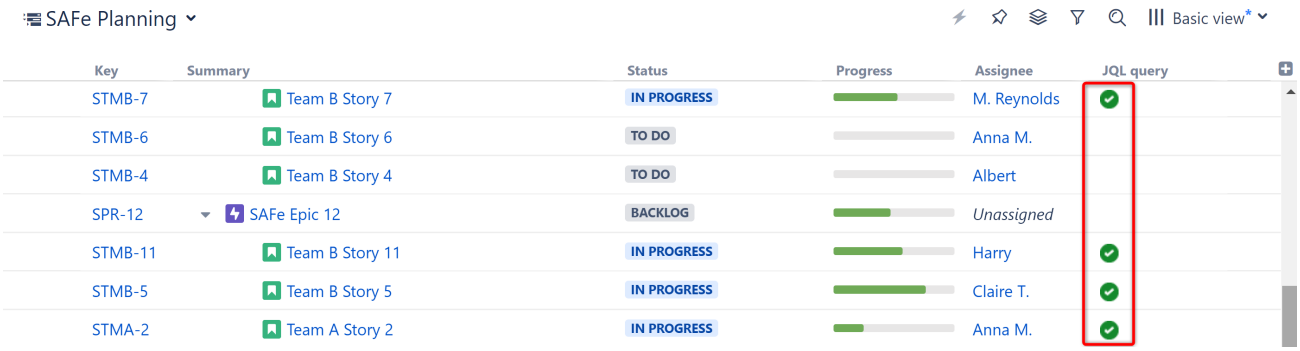
195 http://alm.works/covid19
196 https://demo-structure.almworks.com/

You can also edit text using wiki markup by switching to the Text mode - see Jira's Text Formatting Notation Help page[197] for a complete list of available formatting options and conventions.

Documentation: Memo(see page 97)

7.4.2.2 Query Match Columns

Query Match columns allow you to mark issues that match a specified JQL, S-JQL(see page 266) or text query. Items that match the query are noted by a check mark; those that do not are left blank.

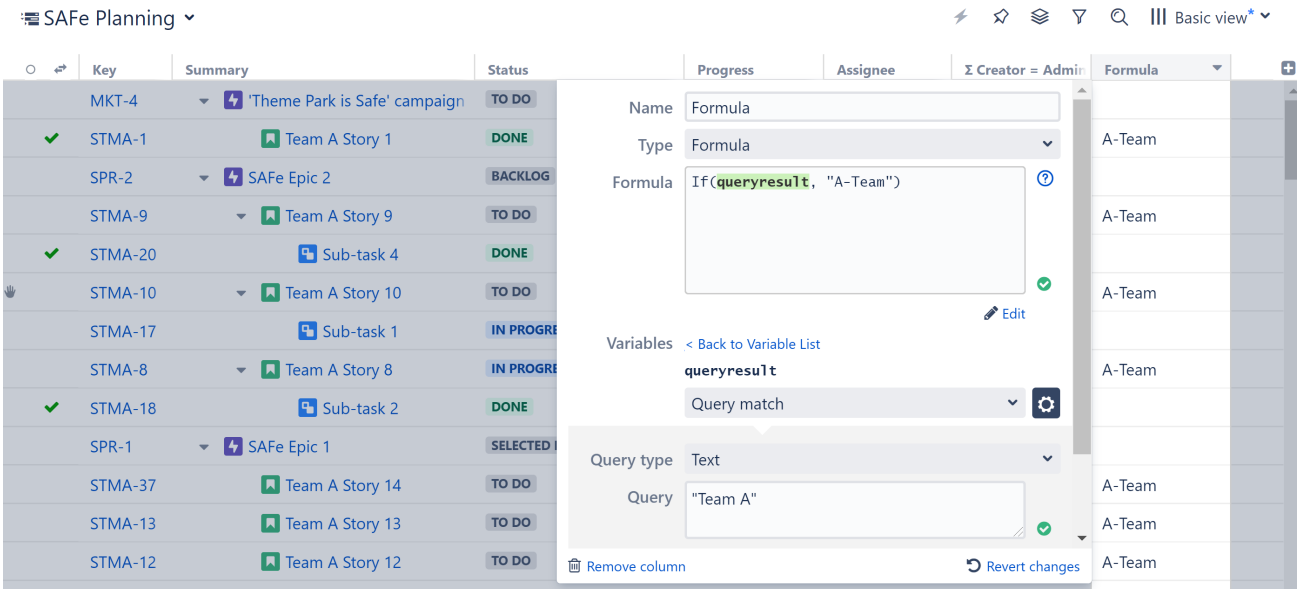


It is also possible to return a numeric value of "1" for any issue matching the query, so you can aggregate your results up the hierarchy.

Documentation: Query Match Column(see page 328)

7.4.2.3 JQL, S-JQL and Text Filters in Formulas

Using Query match, you can now filter formula results based on a JQL, S-JQL(see page 266) or text query.



Documentation: Query Match as Variables(see page 206)

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

7.4.2.4 Jira Anonymization

Structure now supports Jira Anonymization. When an admin anonymizes a user, any private structures, views, Effector processes or statistics owned by the user will be deleted. Shared structures or views owned by the user will be transferred to another user.

Documentation: Anonymizing Users(see page 544)

7.4.2.5 Additional Updates

- Issue summary now appears in the header of the Issue Details panel, so it remains visible as you scroll
- Epic link can now be defined when cloning a structure
- Fixed: the keyboard shortcut "i" (assign to current user) did not work on Agile Boards
- Fixed: Epic Sum Up custom field values were displayed incorrectly within structures

7.4.3 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.4.3.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [198] page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.4.3.2 Upgrading Structure

The upgrade procedure from versions 3.0–6.0 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

198 https://wiki.almworks.com/display/structure2gmaster/Download

7.4.3.3 Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[199] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Structure Quick Start Guide(see page 598).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.4.4 Enterprise Deployment Notes

In this release we have further optimized caching in the attribute system. Now in certain scenarios, when there are many structures that are not frequently used, Structure will clear its attribute caches faster, returning memory to the JVM. There are no particular special areas of interest for load testing and stress testing Structure 6.1. We advise running the same testing procedures as you've done for previous upgrades.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[200].

7.4.5 API Changes in Structure 6.1

7.4.5.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 17.x should work fine.

Jira Version	New API Version
7.13+	17.2.0

See Structure API Versions(see page 883) for full version information and downloads.

199 https://wiki.almworks.com/display/structure2gmaster/Download
200 mailto:support@almworks.com

7.4.5.2 Compatible Changes in the Java API

`StructureDeletedEvent`

Starting with version 6.1, Structure will fire a `StructureDeletedEvent` to Jira's `EventPublisher` whenever a structure is deleted. You can register a listener for this event, for example, if you need to delete any extra data that your code associates with a particular structure.

New utilities

We have added two new classes to the `com.almworks.jira.structure.api.util` package. Both classes are related to integers[201], our primitive collections library:

- `IntegersUtil` contains two methods extracted from `StructureUtil` (to be used in the cloud-based version of Structure).
- `JavaListToLongListAdapter` lets you represent a regular Java `List` as a `com.almworks.integers.LongList` using a `ToLongFunction` for conversion.

7.4.6 Structure 6.1.1 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[202]

17th of June, 2020

Structure 6.1.1 fixes Advanced Wiki Markup and adds Advanced Roadmaps 3.29 support

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[203] (no installation required)

7.4.6.1 Patch Release

Structure 6.1.1 fixes an issue with Advanced Wiki Markup in formulas and descriptions, and adds Advanced Roadmaps (formerly Portfolio) 3.29 support.

7.4.6.2 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

201 https://bitbucket.org/almworks/integers
202 http://alm.works/covid19
203 https://demo-structure.almworks.com/

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [204]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[205] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Structure Quick Start Guide(see page 602).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.4.7 Structure 6.1.2 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[206]

204 https://wiki.almworks.com/display/structure2gmaster/Download
205 https://wiki.almworks.com/display/structure2gmaster/Download
206 http://alm.works/covid19

6th of July, 2020

Structure 6.1.2 fixes an issue with Jira 8.10 compatibility

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[207] (no installation required)

7.4.7.1 Patch Release

Structure 6.1.2 fixes a compatibility issue with Jira Data Center starting from version 8.10. The fix is related to the way Structure issue events are dispatched in Data Center (mostly) and Server. Upon upgrading, Structure will remove the old Structure Index Monitor custom field and create a new global custom field with the same name and description.

Upgrade is highly recommended for Jira 8.10+

7.4.7.2 Installation and Upgrade

Pick a Time

We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [208]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading Structure

The upgrade procedure from versions 3.0–6.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

207 https://demo-structure.almworks.com/
208 https://wiki.almworks.com/display/structure2gmaster/Download

Upgrading from Structure 2.9–2.11

If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[209] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Structure Quick Start Guide(see page 603).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.5 Structure 6.0 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[210]

26th of March, 2020

Structure 6 adds Effectors, user-based permissions, and API changes.

Download the latest version of Structure and its Extensions(see page 978)

Try It: Structure Sandbox Server[211] (no installation required)

Pricing Changes

We are introducing new pricing for Structure Server, effective April 1, 2020. Learn more: Structure for Jira New Server Pricing(see page 615)

7.5.1 Version Highlights

- Effectors - a new form of automation for writing Structure attributes to Jira fields
- Possibility to define permissions per user

209 https://wiki.almworks.com/display/structure2gmaster/Download
210 http://alm.works/covid19
211 https://demo-structure.almworks.com/

- API changes[212] that may affect Structure integrations
- Attribute sensitivity settings
- Improved rank sorting

7.5.2 Changes in Detail

7.5.2.1 Effectors

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

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

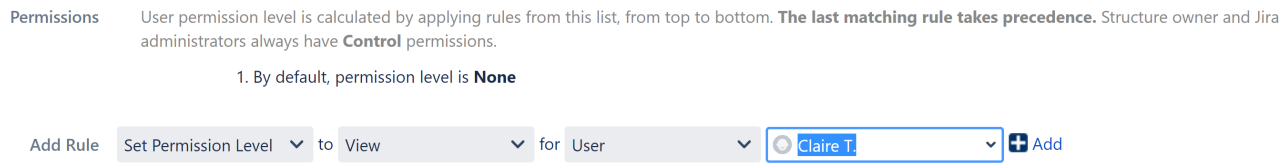
https://www.youtube.com/watch?v=QSreuWUQTp8

For more information, see our Effectors documentation(see page 154) and check out these sample use cases:

- Calculate Story Points for Epics and Write the Values to Jira(see page 433)
- Update Assignees to Match Parent Issues(see page 428)
- Calculate the Cost to Complete Issues and Projects - and Write Those Costs to Jira(see page 423)

7.5.2.2 Permissions Per User

It is now possible to assign structure permissions to specific users. Previously permissions could only be set based on user groups or project teams.



Documentation: Structure Permissions(see page 362)

7.5.2.3 API Changes

Structure 6 introduces significant changes to Java API. (REST API remains backwards compatible.) The changes are called for because of a large remake of the attributes subsystem in the product.

If you have an integration or an extension of Structure, your code might break. More specifically:

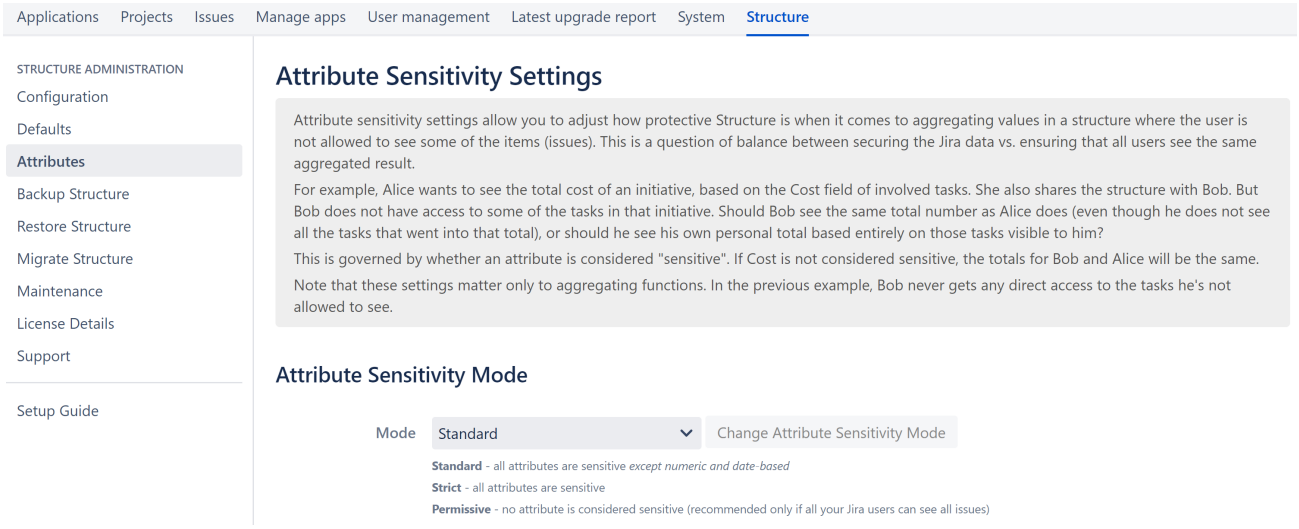
- If you're creating your own attribute loaders, your code will definitely be incompatible and will need some work.
- If you're just using the attribute subsystem, such as `StructureAttributeService`, your code may well break, but fixing it will most likely be trivial.
- If you're using REST APIs or other parts of Structure API, your code is very unlikely to stop working.

212 https://wiki.almworks.com/display/structure2gmaster/Structure+6+API+Changes

For more specifics about these API changes, see Structure 6 API Changes(see page 610).

7.5.2.4 Attribute Sensitivity Settings

Attribute sensitivity settings provide a new level of data security within Structure, allowing admins to specify fields that may contain sensitive information and, therefore, should not be included when calculating aggregated functions.



Documentation: Attribute Sensitivity Settings(see page 483)

7.5.2.5 Improved Rank Sorting

When items are sorted by rank, sub-issues now appear beneath their parent items.

7.5.3 Supported Versions

Structure 6.0 and all extensions support Jira versions 7.13 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

Compatible plugins versons:

- Structure.Gantt[213] 2.3
- Structure.Pages[214], 1.5
- Structure.Testy [215] 2.4

213 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview
214 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
215 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview

7.5.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.5.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download [216]page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.5.4.2 Upgrading Structure

The upgrade procedure from versions 3.0–5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.5.4.3 Upgrading from Structure 2.9–2.11

> If you have a Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

Starting with version 6.0, Structure is no longer able to access the old Structure 2.x database, but old backup files are still supported. Therefore, you'll need to back up your Structure 2.x data *before* upgrading. The recommended upgrade procedure is as follows:

1. While still running the old version of Structure, go to **Administration | Structure | Structure Backup** and create a backup of the current Structure data.
2. Download and install Structure 6.0, either from the Atlassian Marketplace or our Download[217] page.
3. When the Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the Structure Quick Start Guide(see page 605).
4. To transfer your data, go to **Administration | Structure | Restore Structure** and use the Structure 2.x backup made earlier.

---

216 https://wiki.almworks.com/display/structure2gmaster/Download
217 https://wiki.almworks.com/display/structure2gmaster/Download

5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.5.5 Enterprise Deployment Notes

Structure 6.0 is a major release with a lot of changes in the attribute system and an important new feature, effectors.

7.5.5.1 Changes to the Attribute System

In Structure 6.0 we have largely rewritten the attribute system, which is a crucial part of Structure, responsible for displaying all values that you see in columns inside the Structure grid, including issue field values, aggregates, and formulas. We have run extensive performance tests internally, and we are pretty sure that the new attribute system performs no worse than the old one, and significantly better in many cases.

However, we still advise you to perform load- and stress-testing on a staging environment before you upgrade, especially if you rely on aggregates and complex formulas, or export large structures to printable and Excel formats.

7.5.5.2 Effectors

Effectors are a completely new feature, and they shouldn't affect the performance of the rest of the product. However, effectors were a pretty popular feature request, and we expect their wide adoption. If you already have a potential use case for effectors in your organization, we advise you to try them on a staging system before upgrading.

Using effectors can put a noticeable load on the system, because running an effector, especially on a large structure, could require a lot of issues to be loaded from the database, and then updated. Because of that, we designed effectors so that they do not run automatically; instead, they have to be controlled manually by Structure users. Effectors always operate on behalf of the user who runs them, and all the necessary permissions are checked as if that particular user were trying to make the changes.

Access to Effectors

By default, any user who has the "Access Automation" global Structure permission will be able to add effectors to their structures, and any user with the "Bulk Change" global Jira permission will be able to run previously-installed effectors. If you prefer a gradual roll-out, you can limit users' access to effectors as described in Changing Permissions to Configure and Run Effectors(see page 480).

Effector Execution

Running an effector is a two-step process. First, a preview is generated, listing all the changes that the effector is going to make. Then, if the user approves the preview, those changes are actually made, in a separate background process.

To calculate effector previews, Structure uses a separate thread pool (in a Data Center environment, on each node), so several previews can be calculated in parallel. By default, the effector thread pool is limited to N+1 threads, where N is the number of available processor cores. You can change the maximum size of the thread pool by setting the "structure.jobManager.effectorThreads" dark feature.

In contrast to preview calculation, effectors never update issues in parallel. If several users want to make changes using effectors, their requests are queued and processed one-by-one. In a Data Center environment, a single node is responsible for making the changes. If that node leaves the cluster, a different node picks up this responsibility, continues the current task from where it stopped, and takes the next task from the queue.

New Database Tables

There are three new tables in the database schema to store the data related to effectors.

The `AO_8BAD1B_EFFECTOR_INSTANCE` table stores effector parameters. A new row is created each time a user adds an effector to a structure. The data from this table is exported when you back up Structure data and can be migrated to a different instance. The table is also exported along with all Structure data during Jira backup.

The `AO_8BAD1B_EFFECTOR_PROCESS` table stores current and past effector runs. There is one row per process, regardless of how many effectors are run. When a preview is calculated, it is also stored in this table, and deleted when all changes are applied. Effector runs are not backed up by Structure, but the table is exported during Jira backup.

The `AO_8BAD1B_EFFECTOR_RECORD` records all changes made by effectors, and how to undo them. There is one row for each update attempt, successful or not. This data is not backed up by Structure, but the table is exported during Jira backup.

There is a new maintenance task that deletes old effector processes and their changes. By default, finished effector processes and their changes are kept for 30 days, and unfinished processes that didn't make any changes are kept for 24 hours. For more information on automatic maintenance please refer to the documentation(see page 490).

7.5.6 Structure 6 API Changes

7.5.6.1 State of the API

With Structure 6, we are introducing a number of breaking changes to Java API. (REST API remains backwards compatible.) The changes are called for because of a large remake of the attributes subsystem in the product.

If you have an integration or an extension of Structure, your code might break. More specifically:

- If you're creating your own attribute loaders, your code will definitely be incompatible and will need some work.
- If you're just using the attribute subsystem, such as `StructureAttributeService`, your code may well break, but fixing it will most likely be trivial.
- If you're using REST APIs or other parts of Structure API, your code is very unlikely to stop working.

7.5.6.2 Conceptual API Changes

Attribute Sensitivity Configuration

There are new global per-instance settings in Structure, called attribute sensitivity configuration. The administrator has access to them via Administration | Structure | Attributes.

These settings govern the situation where a total or some other type of aggregation is performed over a forest that the user does not have a full access to – there are items that they don't see.

218 mailto:support@almworks.com

StructureConfiguration[219] interface now can be used to read or update these settings with `getAttributeSensitivitySettings()` / `setAttributeSensitivitySettings()`.

Attribute Value Trails Removed

While the concept of trails remains, the API clients no longer have to deal with the trails. The clients are not supposed to implement their own cache – instead, they can request loading as frequently as needed, or use the Attribute Subscription Service.

Asynchronous Attribute Loading

`AttributeSubscriptionService` is now a preferred way to continuously receive updates on the attribute values for a particular set of rows and attributes. It is being used by the Structure grid and through REST APIs.

Support for Super-Root

We have introduced a concept of a "super-root", which is a virtual row above all the roots in the forest. It can be used to calculate a total across the whole forest, for example.

In order to support the super-root we had to reserve `-1` as a special row ID number that identifies the super-root and only the super-root. It's now impossible to use `-1` in Forest structures.

For more details, see `SuperRootRow` class.

7.5.6.3 Detailed API Changes

We list only some of the most important changes in this list. For a detailed description of the current API, see the javadocs(see page 883).

Class Name	Backwards-Incompatible Changes	Other Changes
`AttributeValue`	• Moved to package c.a.j.s.api.attribute. • Trails are no longer a part of the value.	• Stricter contract on nulls is defined.
`StructureAttributeService`	• Method `getAttributeValues()` now returns `RowValues` instead of `VersionedRowValues`. • Method `getAttributeValues()` with both `Forest` and `ForestSpec` parameters is removed.	• `loadAttributeValues()` methods, which push values through a receiver interface, are added. • `getConsistentAttributeValues()` methods are added. • `getItemValues()` method is added – it can load some attributes just for the item IDs, without a forest. • `isItemAttribute()` method helps with figuring out if an attribute is item-based or forest-based.

219 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/settings/StructureConfiguration.html

Class Name	Backwards-Incompatible Changes	Other Changes
`ValueFormat`	• Method `cast(AttributeValue)` is removed.	
`AttributeSpec`	• Method `cast(AttributeLoader)` is removed.	• Attribute normalization rules are updated. Technically, it's a breaking change, but it should not affect custom attributes.
`RowValues`	• `getTrail()` method removed.	• `consume()` method added.
`RestAttributeSpec`	• `fromModel()` method renamed to `toRest()`.	
`ItemForestBuffer`	• Class is removed.	
`MigrationAwareSynchronizer`	• Class is removed.	
`SyncUtil`	• Class is removed.	

7.5.6.4 Conceptual SPI Changes

You only need to bother with SPI if you extend Structure functionality by adding new attributes.

New Attribute Loader Hierarchy

With the new Attributes subsystem we've revamped and extended the types of attribute loaders / attributes that are supported. The new Structure now has support for the following attributes:

- Item attributes are based only on an item and not on its position in the forest.
- Derived attributes are calculated only based on the values of other attributes.
- Single-row attributes are calculated based on the item and the row in the forest that this item is in.
- Multi-row attributes are all based on the dependencies and a particular type of forest-based aggregation.
    - Aggregate attributes are based on the children values of the same attribute.
    - Propagate attributes are based on the ancestor value of the same attribute and sibling rows.
    - Scanning attributes are based on the value of a preceding row, without regard for the hierarchy and when the hierarchy is fully expanded.

For a more in-depth description, see the AttributeLoader javadoc page[220]. Note that there's a parallel hierarchy of `AttributeLoaderContext` classes that has changed.

220 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/loader/AttributeLoader.html

Attribute Caching

The new attribute subsystem is capable of caching attribute values in much more cases than the previous one. It is suggested that the implementations of attribute loaders avoid using `AttributeCachingStrategy.MUST_NOT` and review the `AttributeLoader` interface to see how the caching may be supported, for example, through context dependencies.

Also, we have introduced a new caching strategy – `AttributeCachingStrategy.SHOULD_NOT`, which is handy when the calculated object is heavy and should not be stored, but it allows caching of the dependent attributes.

Context Dependencies

The loaders may now declare their dependencies on some contextual values, for example, on user locale. This governs the caching and invalidation.

On the other hand, in order to call methods like `AttributeLoaderContext.getLocale()`, the loader must declare the context dependency.

Loader Builders

Item and single row loaders can now be built more conveniently with a number of builders. Start with `AttributeLoaders` class.

7.5.6.5 Detailed SPI Changes

We list only some of the most important changes in this list. For a detailed description of the current API, see the javadocs(see page 883).

Class Name	Backwards-Incompatible Changes	Other Changes
`AttributeContext`	• Replaced method `getBaseForestSpec()` with `getBaseStructureId()`	• Added `getI18n()` method • Added `getTimezone()` method
`AttributeLoader`	• Replaced the sub-interfaces and their methods, including how values are loaded	
`AttributeLoaderProvider`	• `createAttributeLoader` method now accepts `AttributeProviderContext` instead of `AttributeContext`	
`BulkAttributeLoader`	• `BulkAttributeLoader` interface is eliminated. Every loader (except DerivedAttributeLoader) is now a bulk-loading loader.	

Class Name	Backwards-Incompatible Changes	Other Changes
`AbstractDistinctAggregateLoader`	• Renamed to `AbstractNaiveDistinctAggregateLoader`	
`AttributeLoaderSecurity`	• Class removed: there's no need in supporting value security in the individual loaders anymore – the system does it by itself, with the sensitivity settings governing the multi-row loaders.	
`AbstractDistinctSumLoader`	• Distinct sum loaders are currently removed from the API.	
`SimpleAttributeProvider`	• Moved to package `c.a.j.s.api.attribute.loader.basic`.	
`TrailItemSet`	• Moved to package `c.a.j.s.api.attribute.loader`.	

7.5.6.6 Effector APIs and SPIs

Structure 6.0 also adds a set of new APIs and SPIs related to effectors. This section only provides a basic overview, for details please refer to the Javadoc[221].

Managing Effectors

You need the `EffectorInstanceManager`[222] to create, retrieve, update, and delete *effector instances*. To create an effector instance, you must provide a complete module key of an effector implementation and a map of parameters. Once you have the effector instance ID, pass it to `CoreIdentities.effector(long)`[223] to get an item identity, which you'll be able to insert into a structure.

As of version 6.0, Structure provides only one effector implementation, the Attribute to Issue Field(see page 154) effector. Its module key is "com.almworks.jira.structure:effector-attribute-to-field".

Running Effectors

You need the `EffectorProcessManager`[224] to run effectors. You start an *effector process* by calling `startEffectorProcess()`, which returns a process ID. You then use this process ID to track the status of your process, confirm effect application, and examine the results.

---

221 https://almworks.com/structure/javadoc/latest/
222 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/instance/EffectorInstanceManager.html
223 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/CoreIdentities.html
224 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/process/EffectorProcessManager.html

For a finished process, you can also start a new process that would undo the changes it made.

Writing a New Effector

As a Jira plugin developer, you can extend Structure with new effectors. Structure 6.0 adds two new extension points, *effectors* and *effect providers*. Effectors look at a forest and attribute values to generate *effect descriptions* (represented by `StoredEffect`[225]). Effect providers validate effect descriptions and perform the requested changes.

This architecture supports code reuse. For example, Structure itself knows how to properly update most Jira issue fields, and supplies corresponding effect providers. So if all your effector does is update issue fields, you can use those providers and avoid writing issue-changing code yourself.

Effectors implement the `Effector`[226] interface and are registered using the `<structure-effector>` (see page 893) module type. Your effector implementations can use the static factory methods in the `CoreEffects`[227] class to generate effect descriptions that will be handled by Structure.

Effect providers implement the `EffectProvider`[228] interface and are registered using the `<structure-effect-provider>` (see page 894) module type.

7.5.7 Structure for Jira New Server Pricing

We are introducing new pricing for Structure for Jira, effective April 1, 2020. These changes will only affect Server licenses. Data Center and Cloud pricing will remain the same.

To make the transition easier, there will be a 3-month grandfathering period. Until June 30, 2020, you will be able to purchase your renewal or a new license at the old price. To take advantage of this option, please contact our support team at support@almworks.com[229] or create a request through our customer portal: https://support.almworks.com/servicedesk/customer/portal/8.

See below for a detailed breakdown of the new pricing. All prices are in US Dollars.

User Tier	Current	New
10	$10	**$10**
25	$600	**$720**
50	$1100	**$1320**
100	$2000	**$2400**
250	$4000	**$4800**
500	$6000	**$7200**
2000	$8000	**$9600**

225 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/StoredEffect.html
226 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/Effector.html
227 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/CoreEffects.html
228 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/EffectProvider.html
229 mailto:support@almworks.com

User Tier	Current	New
10000	$12000	**$15600**
10000+	$16000	**$20800**

7.5.8 Structure 6.0.1 Release Notes

ALM Works Business Continuity During the COVID-19 Outbreak[230]

5th of May 2020

Structure 6.0.1 provides performance and security related fixes

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[231]

7.5.8.1 Patch Release

This is a small patch release based on Structure 6.0. It provides performance and security related fixes.

Upgrade is recommended for all customers using Structure 6.0.

7.5.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.5.8.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 6.0.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.

230 http://alm.works/covid19
231 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.5.8.4 Enterprise Deployment Notes

Structure 6.0.1 introduces attribute caching optimizations for structures with pinned items.

This is designed to reduce memory consumption for structures which are loaded on Issue View pages because they contain pinned issues. Since structures are not shown on the Issue View pages of issues added through Automation, this is particularly important for instances with huge structures built by synchronizers or containing large amounts of static content.

7.6 Structure 5.6 Release Notes

> **19th of September, 2019**
>
> Structure 5.6 adds column pinning, cache clearing, and more.

Download the latest version of Structure and its Extensions
(see page 978)Try It: Structure Sandbox Server[232] (no installation required)

7.6.1 Version Highlights

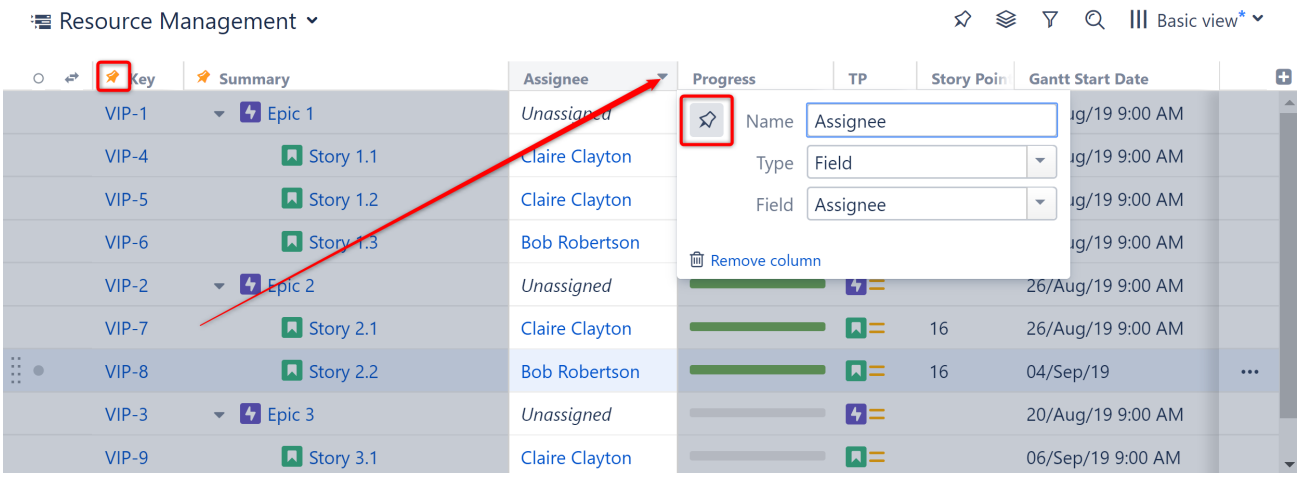
- Columns can be "pinned" to remain visible when horizontal scrolling is enabled.
- Structure caches can now be cleared for users and nodes from within the app.
- Adds support for Jira 8.4

7.6.2 Changes in Detail

7.6.2.1 Column Pinning

When Horizontal scrolling is enabled, the Summary column is "pinned" so that it will always be visible as users scroll through other columns. Now users can pin additional columns, so they will remain visible as well.

232 https://demo-structure.almworks.com/

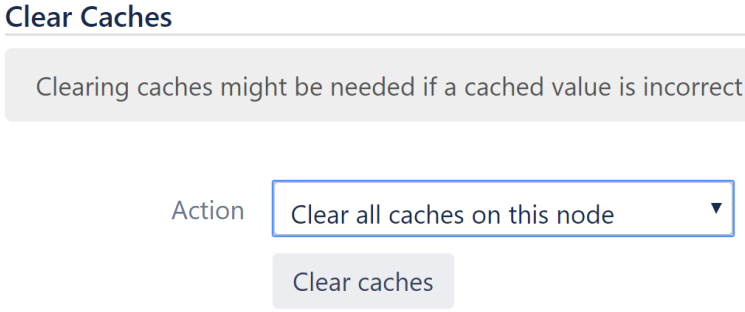


Pinned columns will be moved to the left side of the screen and will remain visible as you scroll.

Documentation: Horizontal Scrolling(see page 333)

7.6.2.2 **Clearing Structure Caches**

Structure caches can now be cleared from within the app for specific users or the whole Jira instance. In Jira Data Center, it's possible to clear caches for the current node or all nodes in the cluster.



Documentation: Clearing Structure Caches(see page 543)

7.6.2.3 Additional Updates

- Archived structures are now clearly marked as archived when opened.
- When viewing a structure on a Project page, the current project is auto-selected when adding a new issue.
- Fixed: Compatibility with Jira add-ons, "Links Hierarchy - Core, Epic & Portfolio" and "Tree CustomField"

7.6.3 Supported Versions

Structure 5.6 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.5. Structure.Testy[233]extension, Colors[234], Structure.Pages[235], Structure.Gantt[236]and integrations with third-party apps should continue working normally.

7.6.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.6.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 978) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.6.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure from versions 3.x–5.5 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.6.5 Enterprise Deployment Notes

Structure 5.6 contains a workaround for a performance problem that could affect extremely large Jira 7 instances, and a few other changes in important areas such as client-server communication, structure generation and locking, and user action handling.

233 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
234 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
235 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
236 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

7.6.5.1 Lucene Searcher Flushing

Jira uses Apache Lucene as the search engine behind JQL, and so Structure also relies on Lucene for JQL and a variety of other tasks, such as checking users' access to issues and reading issue data quickly. To work with Lucene, an app uses a Lucene searcher object provided by Jira. Lucene searchers are shared between Jira itself and all apps; when an issue is updated and re-indexed, the current Lucene searcher is released, and it can be garbage-collected when all tasks still using it are finished.

In Jira 7, on extremely large instances, Lucene searchers and associated data structures can consume huge amounts of memory – see JRASERVER-67805[237], JRASERVER-68439[238]. This shouldn't affect Jira 8, which uses a newer version of Lucene, where these problems are fixed.

We have seen a few support cases, with very large Jira 7 instances, where a long-running Structure task would hold on to a Lucene searcher that consumes several gigabytes of memory. This would prevent the JVM from garbage-collecting the searcher and reclaiming that memory, which in turn would negatively affect the performance of the whole instance or cluster node. Note that Structure itself does not necessarily *need* all that memory; the shared searcher might get inflated by an unrelated task and *then* be given to Structure.

Lucene searcher flushing is a new, experimental dark feature introduced to work around this problem. If it's enabled, Structure will release ("flush") the Lucene searcher not at the end of a task, but immediately after using it. The JVM will then have a chance to garbage-collect the searcher without waiting for the current Structure task to finish.

Lucene searcher flushing is available only on Jira 7. It is enabled automatically for instances where the number of issues in the index multiplied by the number of searchable fields is greater than or equal to 2^{30} (1,073,741,824). It can also be enabled or disabled manually by setting the `structure.lucene.flushSearchers` dark feature to "true" or "false", respectively. If you are running Jira 7, and your instance is above the threshold or close to it, and you have large structures that take a long time to generate, we advise that you load- and stress-test your instance before upgrading.

7.6.5.2 Reduced HTTP Thread Pool Usage

Structure does not perform potentially costly operations like structure generation in HTTP threads, offloading them to background thread pools instead. In previous versions of Structure, an HTTP thread would start a background job, wait for it to finish, and if it's not done in 10 seconds, return the job ID to the client. However, the client would immediately re-submit the request, and thus it would effectively always keep one HTTP thread busy for the duration of the background task.

As we have observed in a recent support case, on an overloaded Jira instance, Structure's background tasks could also slow down, and, in extreme cases, too many HTTP threads would be busy waiting for Structure tasks, making it hard for the instance to serve other HTTP requests.

To reduce chances of HTTP thread pool starvation, Structure 5.6 introduces client-side HTTP timeouts. When the client receives a job ID, it will wait 1 second before re-submitting the request, then 2 seconds, 4 seconds, and so on, up to 1 minute. We have also reduced the server-side timeout from 10 seconds to 5 seconds, so HTTP threads are returned to the pool sooner, improving the throughput.

237 https://jira.atlassian.com/browse/JRASERVER-67805
238 https://jira.atlassian.com/browse/JRASERVER-68439

7.6.5.3 Clearing Caches and Hard Refresh

In previous versions, hard-refreshing a page in the browser would cause Structure to clear its caches for the current user. We have decided to change this for two reasons. First, clearing the caches too often could negatively affect performance. Second, some of Structure's important caches are global, not per-user, and there was no easy way to clear them. Structure 5.6 introduces a new way of clearing caches. Hard-refreshing the page no longer works, so if you relied on it for troubleshooting Structure issues, please switch to the new interface(see page 543).

7.6.5.4 Testing on Staging Environment

We advise you to perform load and stress testing in a staging environment before you upgrade. Given the changes in this version, you can try the following scenarios:

- Find or create a large automated structure that takes significant time to generate (around 30 seconds or more). Emulate multiple users opening it at the same time.
- Emulate multiple users making changes to Jira issues in this structure at a significant rate (say, one update every 1 or 2 seconds). Check that the structure re-generates to reflect the changes.
- Open a large manually-created structure containing 10,000 issues or more. Try moving one or more rows in this structure, note the time it takes to handle the moves.

While running these experiments, watch the log files for errors and warnings, and monitor heap usage and GC activity.

> Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[239].

7.6.6 API Changes in Structure 5.6

7.6.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.16.0

See Structure API Versions(see page 883) for full version information and downloads.

7.6.6.2 Compatible Changes in the Java API

Pinned columns in `ViewSpecification`

With the introduction of pinned columns in Structure 5.6, we have added a list of pinned column IDs to the view specification. There are 3 new methods to work with the list:

239 mailto:support@almworks.com

- `ViewSpecification.getPins()`
- `ViewSpecification.Builder.setPins(List<String>)`
- `ViewSpecification.Builder.getPins()`

Resetting the `ItemTracker`

We have added a new method named `reset()` to the `ItemTracker` interface. It's meant to be used in situations when there are too many changes, or when it's impossible to know which items have changed. All clients calling `getUpdate()` after the `reset()` call will receive full updates.

7.6.7 Structure 5.6.1 Release Notes

14th of October 2019

Structure 5.6.1 fixes a script dependency that may cause the Issue Navigator page to load incorrectly.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[240]

7.6.7.1 Patch Release

This is a small patch release based on Structure 5.6.0.

Upgrade is recommended for all customers using Structure 5.6.0.

7.6.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.6.7.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.

240 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.6.8 Structure 5.6.2 Release Notes

24th of December 2019

Structure 5.6.2 provides Jira 8.4 compatibility fixes and more.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[241]

7.6.8.1 Patch Release

This is a patch release based on Structure 5.6.1. It provides:

- Compatibility with Jira 8.4+.
- Fixed: Epic Link colors now appear correctly in gadget.
- Fixed: “Archived” label appeared next to search type when a search is opened.
- Fixed: After collapsing and expanding structure levels, user could not select status for the Structure.Testy add-on.
- Additional bug fixes.

Upgrade is recommended for all customers using Structure 5.6.x

7.6.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.6.8.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

241 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.6.8.4 4. Enterprise Deployment Notes

Structure 5.6.2 does not introduce changes that could affect performance or would justify additional load and stress testing for Enterprise deployments.

7.6.9 Structure 5.6.3 Release Notes

> **24th of January 2020**
>
> Structure 5.6.3 provides performance improvements for Jira 8+ and the Portfolio Extender.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[242]

7.6.9.1 Patch Release

This is a patch release based on Structure 5.6.2. It provides performance improvements for Jira 8+ and the Portfolio Extender.

Upgrade is recommended for all customers using Structure 5.6.x.

7.6.9.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.6.9.3 Upgrade

> If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

242 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.6.9.4 4. Enterprise Deployment Notes

Structure 5.6.3 significantly increases Lucene index reading operation performance on Jira 8+. The new version of Lucene in Jira 8 introduced major changes to index data structures, which made its traditional read APIs significantly slower on large amounts of data. Structure 5.6.3 compensates for this, providing overall performance on Jira 8.x comparable with Jira 7.x, or even faster. We have also improved the performance of the Portfolio children extender, which now batches Lucene queries in some cases and should work faster on Jira 8+. We advise you to run Structure 5.6.3 tests on your slowest structures pre-update (for example, structures using the Child Issues (Portfolio) Extender) and compare the new times against your current Structure version.

7.6.10 Structure 5.6.4 Release Notes

> **5th of May 2020**
>
> Structure 5.6.4 is a security patch release.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[243]

7.6.10.1 Patch Release

This is a patch release based on Structure 5.6.3. We have addressed a security issue that affects all Structure versions starting with 3.5. Problem was fixed in the latest (6.0) Structure.

Upgrade is recommended for all customers using Structure 5.6.x.

7.6.10.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.6.10.3 Upgrade

> If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 5.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)

243 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.7 Structure 5.5 Release Notes

18th of July, 2019

Structure 5.5 adds performance safeguards for structure transformations and contains a few improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 978)Try It: Structure Sandbox Server[244] (no installation required)

7.7.1 Version Highlights

- Timeout for transformations
- Possibility to back up and restore Structure configuration
- Descending order is set by default when sorting by version
- Manual adjustments improvements and bug fixes

7.7.2 Changes in Detail

7.7.2.1 Transformation Timeout

In Structure 4.6, we introduced a feature for pausing the structure generation(see page 172) process if it takes too much time. If loading a structure with automation takes longer than allowed, the generation process would stop to avoid performance impact on Jira. This limit was not applied to transformations, which could potentially cause the same performance impact.

In this version we've introduced a mechanism that will pause the generation process if it exceeds the time limit, regardless of whether it is caused by the automation that is a part of the structure, or by the additional transformations applied on top.

7.7.2.2 **Backup/Restore/Migration Improvements**

We have added several additional options for Structure Backup/Restore/Migration operations. Starting with Structure 5.5, the Structure configuration and the dark feature settings(see page 499) will be stored not only in Jira backups but also in Structure backups. When restoring from a Structure backup, the saved Structure configuration (like the settings set by the Jira administrator) will overwrite the active Structure configuration.

When migrating Structure data, you can choose if you want to restore the general Structure permissions settings(see page 478), such as projects enabled for Structure, and if you want to restore the enabled dark features.

244 https://demo-structure.almworks.com/

7.7.3 Supported Versions

Structure 5.5 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.4. Structure.Testy[245]extension, Colors[246], Structure.Pages[247], Structure.Gantt[248]and integrations with third-party apps should continue working normally.

7.7.4 Installation and Upgrade

> **Pick a Time**
>
> We strongly recommend that you install and upgrade your apps during off-peak hours or scheduled maintenance windows. There are known issues in the Jira plugin infrastructure that may cause performance degradation and impede app installation when your Jira instance is under heavy load.

7.7.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 978) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.7.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure from versions 3.x–5.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

245 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
246 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
247 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
248 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

7.7.5 Enterprise Deployment Notes

In terms of stability and performance, this release does not bring significant changes compared to version 5.4. There are no particular special areas of interest for load testing and stress testing Structure 5.5. We advise running the same testing procedures as you've done for previous upgrades.

> Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[249].

7.7.6 API Changes in Structure 5.5

7.7.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.15.0

See Structure API Versions(see page 883) for full version information and downloads.

7.7.6.2 Compatible Changes in the Java API

Restoring History

Restoring Structure history from a backup file can take a long time. Now you can call `BackupOperation.setRestoreHistory()` to disable restoring history even if it's present in the backup file. The new method must be called before calling `restore()`.

Pinger

`Pinger` is an `@Internal` Structure component extracted into the API for consumption by Structure add-ons. It is not intended to be used by third-party developers.

7.7.7 Structure 5.5.1 Release Notes

> Please note that if you are using Jira Data Center after September 02, 2019, you will need to switch to the Data Center license for Structure as well (no changes to the app itself are required).

249 mailto:support@almworks.com

15th of August 2019

Structure 5.5.1 provides CDN support[250] (Jira 8.3 compatibility)

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[251]

7.7.7.1 Patch Release

This is a small patch release based on Structure 5.5.0.

Upgrade is recommended for all Data Center customers using Structure 5.5.0.

7.7.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.7.7.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 5.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.8 Structure 5.4 Release Notes

5th of June, 2019

Structure 5.4 adds Memo items, Status category column, the ability to track time in status and additional improvements and fixes.

250 https://confluence.atlassian.com/adminjiraserver/use-a-cdn-with-atlassian-data-center-applications-974378840.html
251 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Download the latest version of Structure and its Extensions
(see page 978)Try It: Structure Sandbox Server[252] (no installation required)

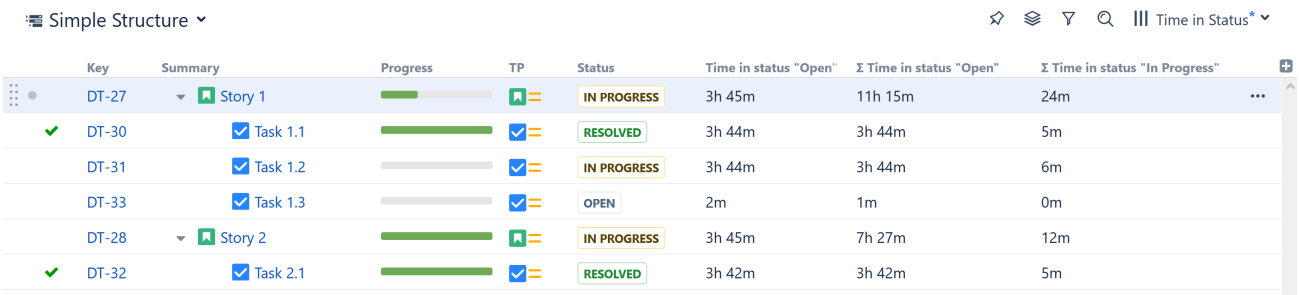
7.8.1 Version Highlights

- Introduces the new Memo item, which allows you to add text notes or flags as rows within a structure
- Introduces the Status category column and the ability to use Status category in formulas
- The ability to track the time issues spend in a particular status
- Additional performance improvements and bug fixes

7.8.2 Changes in Detail

7.8.2.1 Time in Status

The Time in Status column allows you to calculate how much time issues spend in specific statuses.



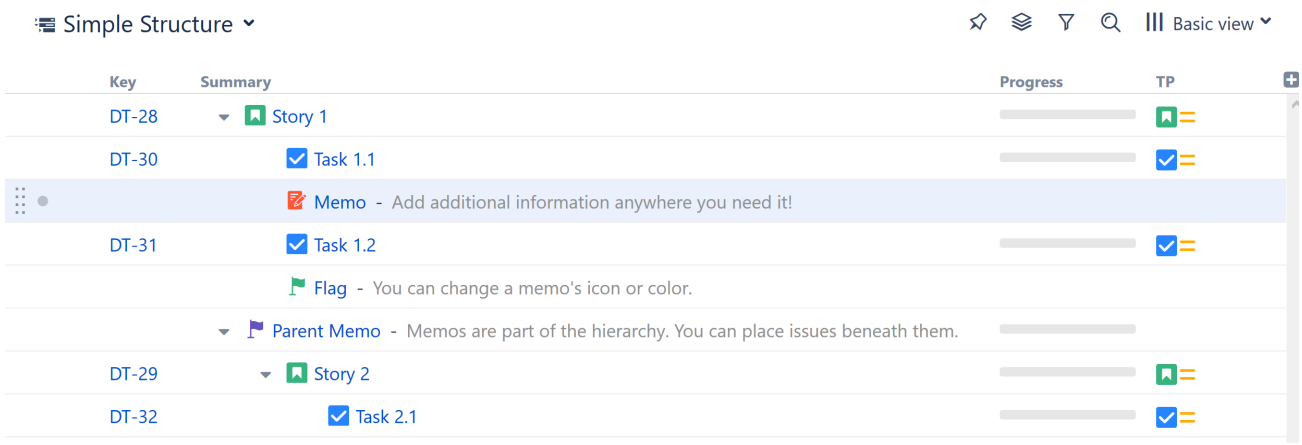
Time in Status can be customized and aggregated, depending on your business needs.

Documentation: Time in Status Column(see page 321)

7.8.2.2 Memo Items

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

252 https://demo-structure.almworks.com/



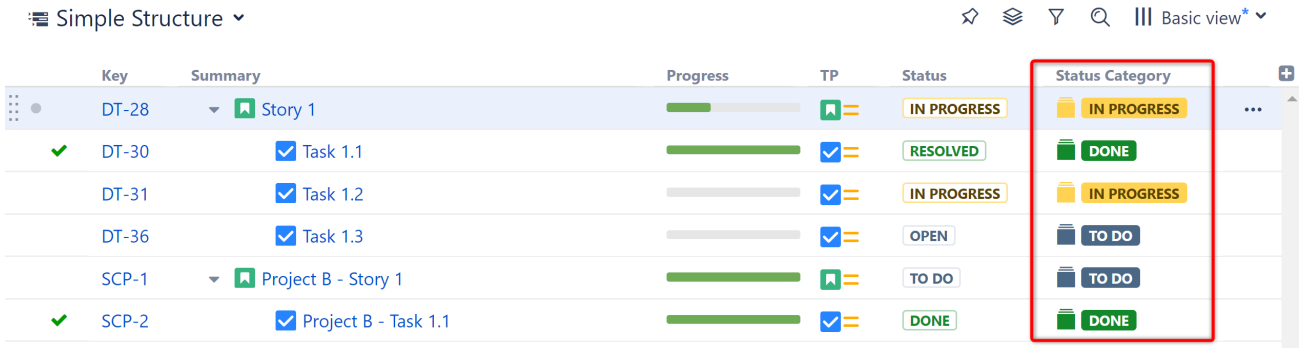
Memos can serve a variety of purposes within a structure:

- Add notes or reminders that pertain to the structure or project as a whole, rather than just a single issue (for that, try a Notes column(see page 319))
- Add high-level requirements directly to your structure
- Add a placeholder for other items
- Use them like folders, grouping issues within your hierarchy (with the added benefit of color and text)
- Just about anything else you can think of!

Documentation: Memo(see page 97)

7.8.2.3 Status Category Column

The Status category column allows you to see at a glance which Status category each issue is in. This can be extremely useful if different teams/projects use different workflows or custom statuses.



Status category values can be used to sort, filter or group issues. They can also be used in JQL queries and formulas.

Documentation: Status Category Column(see page 325)

7.8.2.4 Additional Updates

- Export to Structure has been removed from the Issue Navigator page
- Fixed: With Group by Status generators, unable to move issues within the same Status grouping
- Fixed: Summary value looks empty when editing another field
- Fixed: Values in aggregate columns (with Sum over sub-items option checked) cannot be edited after a sort

- Fixed: Summary is editable inline even if it has been removed from the edit screen
- Fixed: Power Scripts 4.6.0 breaks Structure transformations
- Fixed: Items sometime move incorrectly when Manual Adjustments are used with Automations

7.8.3 Supported Versions

Structure 5.4 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.3. Structure.Testy[253] extension, Colors[254], Structure.Pages[255], Structure.Gantt[256] and integrations with third-party apps should continue working normally.

7.8.4 Installation and Upgrade

7.8.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 978) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on the Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please note that there are limitations to its compatibility with Structure 4.2 and higher, so a Confluence upgrade to version 6.1 or later is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.8.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure from versions 3.x–5.3 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

253 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
254 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
255 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
256 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

7.8.5 Enterprise Deployment Notes

Structure 5.4 introduces two new features and a performance improvement important for large-scale Jira Server and Jira Data Center instances.

7.8.5.1 Periodical Cleaning of the JQL Query Literals Cache

Jira has quite a lot of request-level caches that keep expensive data easily accessible for the duration of a single user request. These caches get cleared once the request completes, and since the absolute majority of requests are quick, there is usually no problem.

However, we have seen a few support cases in which an unusually long-running background task would make a Structure thread accumulate too much data in request caches, consuming excessive amounts of memory.

We are actively working to solve this problem, and in Structure 5.4 we introduce periodical cleaning of the JQL query literals cache, which is one of the caches susceptible to the problem. Further improvements in this area are planned for future Structure releases.

If you have large structures that take significant time to generate, we advise that you check their performance in a staging environment before upgrading.

7.8.5.2 Time in Status Column

The new Time in Status column is based on issue history data, so the values can be quite expensive to calculate. Our performance tests have shown that in the aggregate mode, it can be visibly slower than simple field-based columns in structures of 10,000 issues or more. If you are planning to use this column with large, deeply-nested structures, we advise that you check its performance on a staging system.

7.8.5.3 Memo Item Type

The new Memo item type is based on generic items, introduced in Structure 4.4. The data will be stored in the `AO_8BAD1B_GENERIC_ITEM` table, which will contain one row for each memo item. Because the table has been introduced earlier, there are no schema updates in this release.

Memos may be slower when accessing structure via Chrome with the built-in spell checker enabled.

7.8.5.4 Testing on a Staging Environment

Apart from the suggestions above, there are no special areas of interest for load testing and stress testing Structure 5.4. We advise running the same testing procedures as you've done for previous upgrades.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[257].

7.8.6 API Changes in Structure 5.4

7.8.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.14.0

See Structure API Versions(see page 883) for full version information and downloads.

7.8.6.2 Compatible Changes in the Java API

Memo Item Type

With the introduction of memo items in Structure 5.4, we are adding a new constant to the public API: `CoreItemType.MEMO` contains the module key for the memo item type.

Memo items were called "notes" during development, but we decided to rename them to avoid confusion with the Note column. The constant `CoreItemTypes.NOTE`, added in the previous API version, has therefore been deprecated and shouldn't be used.

New Methods in `TrailItemSet`

We have added two new methods to the `TrailItemSet` class:

- `fromValues()` is a static factory method that lets you combine the trails from a collection of `AttributeValues` into a `TrailItemSet`;
- `expand(TrailItemSet)` produces a set that contains all items from the receiver and the argument sets.

7.8.7 Structure 5.4.1 Release Notes

5th of July 2019

Structure 5.4.1 provides a number of minor fixes.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[258]

257 mailto:support@almworks.com
258 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.8.7.1 Patch Release

This is a small patch release based on Structure 5.4.0. It addresses the following issues:

- Fixed: Screen remains dark after making a status change in the Issue Details panel.
- Fixed: Empty rows appear in the structure when using a combination of Filter and Group transformations.
- Fixed: Summary data is lost when moving from the Create Issue dialogue to the Create Issue panel.
- Fixed: Cannot share Structure.Gantt chart perspectives.

Upgrade is recommended for all customers using Structure 5.4.0.

7.8.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.8.7.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 5.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.9 Structure 5.3 Release Notes

26th of March, 2019

Structure 5.3 adds possibility to group by fix version name and several improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 978)Try It: Structure Sandbox Server[259] (no installation required)

7.9.1 Version Highlights

- Introduces the new Version Name Grouper, which groups issues by version, even across multiple projects

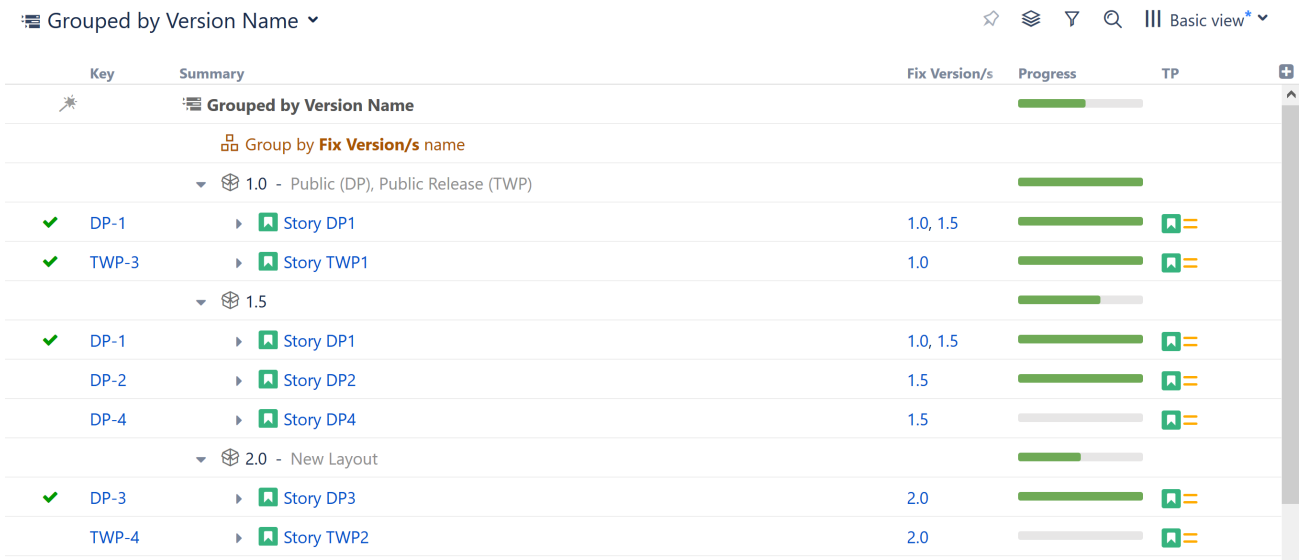
259 https://demo-structure.almworks.com/

- Resolved icons (green checkmarks) are now displayed when an issue is in a Done status category, rather than anytime there is a non-blank resolution
- "Mark Manual Adjustments" option is now switched on by default when Manual Adjustments(see page 175) are enabled
- Quick Transformations panel can now be hidden while transformations remain active
- Fixed: Unable to change epic links with synchronizers installed

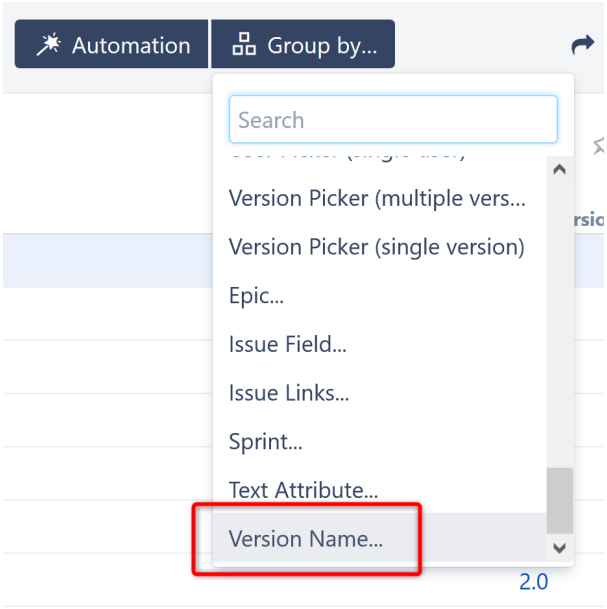
7.9.2 Changes in Detail

7.9.2.1 Version Name Grouper

The new **Version Name...** Grouper allows you to group issues with the same version names across multiple projects.



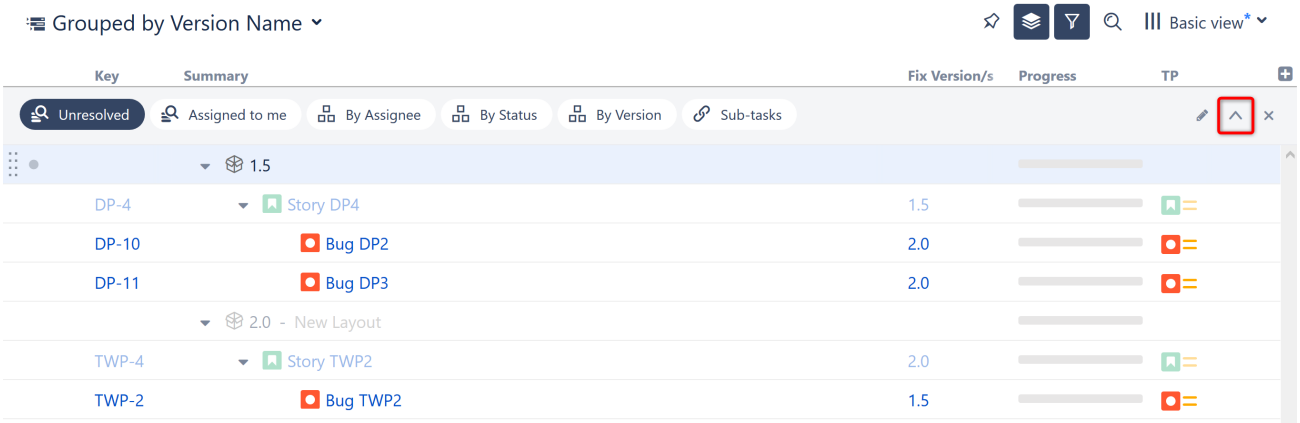
The Version Name Grouper is located at the bottom of the Group by list - or Just type "version name" into the search bar.



You can group issues by the Affects Version, Fix Version or a version custom field (multiple or single).

7.9.2.2 Hide Quick Transformations panel

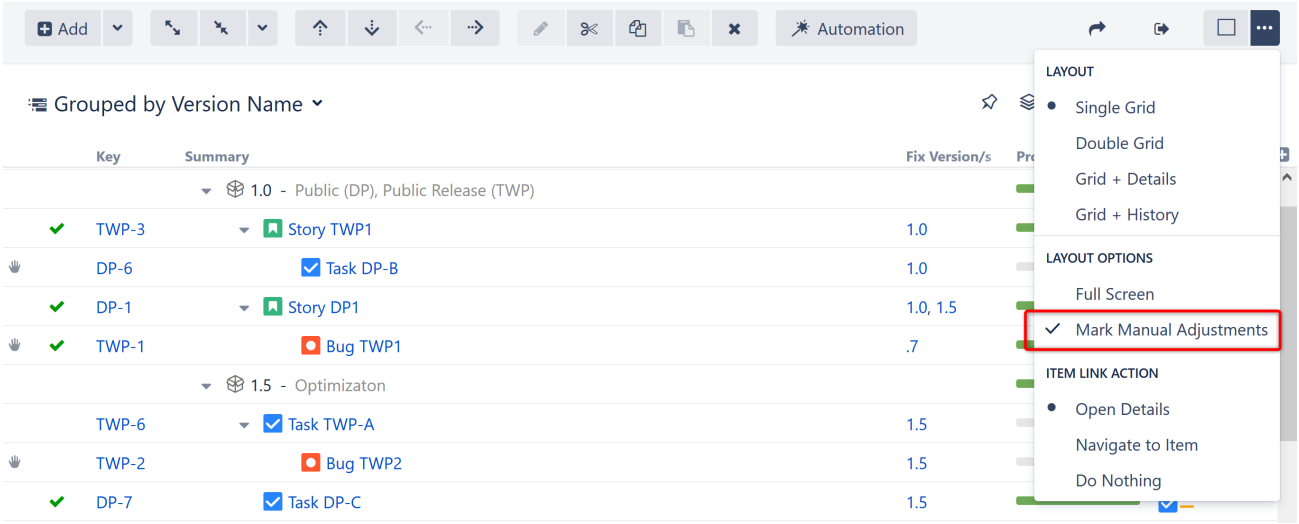
It is now possible to hide the Quick Transformations panel (as could previously only be done with the Transformations panel), while continuing to use Quick Transformations in your structure. Just click the hide button:



When transformations are applied, but the panel is hidden, the Quick Transformations button is colored blue.

7.9.2.3 Mark Manual Adjustments

The Mark Manual Adjustments option is now selected by default, whenever manual adjustments are enabled for a structure.



7.9.2.4 Additional Updates

- Resolved icons (green checkmarks) are now displayed when an issue is in a Done status category, rather than anytime there is a non-blank resolution. Admins can revert this to the old behavior using Advanced Configuration(see page 635).
- Fixed: Unable to change epic links with synchronizers installed.

7.9.3 Supported Versions

Structure 5.3 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.2. Structure.Testy[260] extension, Colors[261], Structure.Pages[262], Structure.Gantt[263] and integrations with third-party apps should continue working normally.

7.9.4 Installation and Upgrade

7.9.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 978) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.

260 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
261 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
262 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
263 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on the Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please note that there are limitations to its compatibility with Structure 4.2 and higher, so a Confluence upgrade to version 6.1 or later is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.9.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure from versions 3.x–5.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

> We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

7.9.5 Enterprise Deployment Notes

Structure 5.3 contains several changes to its core components that could be important for large installations and Jira Data Center instances.

7.9.5.1 Core Attribute Changes

The newly-introduced Version Name item type is special because its core attributes, like summary and icon, depend not on the Version Name item itself, but on the issues below it. In order to support that, we had to change how those attributes are loaded for all item types, not only Version Names.

7.9.5.2 Custom Field Scope Checks

Previous versions of Structure could show custom field values for fields that are no longer available for the project and type of the issue. Those values could also be used by generators, even though Jira itself wouldn't show or let you edit them. Starting with Structure 5.3, we check the scope of custom fields when generating structures and loading issue attributes. We use the Lucene index when possible to speed up these checks.

7.9.5.3 Testing on Staging Environment

Given the changes described above, if you have large, multi-level structures (10,000 issues or more), we advise that you test their performance on a staging system before upgrading. You can try the following scenarios:

- Open a large, multi-level structure using a view with only a few simple columns, e.g. Key, Summary, and Assignee. Note how long it takes for the field values to appear.

- Scroll down a few screens and note how quickly Structure loads field values for the new rows.
- Turn on Automation mode and see how long it takes to render the structure and generator rows.
- Add a custom field column to the Structure grid, and check how long it takes to load the values. Scroll a few screens up or down again.
- Create a large structure with a few generators based on Jira's built-in custom fields, e.g. a grouper by a User Picker field and a sorter by a Number field. Check how long it takes to generate.
- Edit an issue from the structure and change its custom field value. Note the structure's reaction time.

Watch the log files for errors and warnings while running these experiments.

The usual load and stress testing can also be performed.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[264].

7.9.6 API Changes in Structure 5.3

7.9.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.13.0

See Structure API Versions(see page 883) for full version information and downloads.

7.9.6.2 Compatible Changes in the Java API

Version Name Items

In Structure 5.3 we introduce a new core item type, the Version Name. It represents the canonical (lower-case, trimmed) name of a Jira version, and is used by the new Version Name grouper to combine same-named versions from multiple projects. We have added the following static methods and constants to the API to support Version Name items:

- `CoreIdentities.versionName(String)`
- `CoreIdentities.versionName(Version)`
- `CoreIdentities.isVersionName(ItemIdentity)`
- `CoreIdentities.canonicalVersionName(String)`
- `CoreItemTypes.VERSION_NAME`
- `JiraFunc.CANONICAL_VERSION_NAME`
- `JiraFunc.PROJECTCONSTANT_DESCRIPTION`

264 mailto:support@almworks.com

Arbitrary Parameters for Generic Items

We have added a parameter map to the `GenericItem` class (available since Structure 4.5). Parameters are stored in the database, and let you associate arbitrary information with a generic item. The following methods have been added to the API:

- `GenericItem.getParameters()`
- `GenericItem.Builder.setParameter(String, Object)`
- `GenericItem.Builder.setParameters(Map<String, Object>)`

We are currently working on a new item type, the Note, which will be based on generic items. There is a new constant for it, `CoreItemTypes.NOTE`.

Deprecated Methods in Export API

The following methods in the Export API have been deprecated:

- `ExportRequestContext.requireAttribute(AttributeSpec, boolean)`
- `ExportRow.get(AttributeSpec, boolean)`

Starting with version 5.3, Structure handles the `afterFiltering` flag for you, so you should switch to the single-argument versions of those methods.

Custom Final Step for New Structure Templates

It is now possible to skip the common final step in a new structure template wizard, if your template can automatically set the new structure name and permissions. A new default method and a constant have been added to support that:

- `NewStructureTemplate.hasCustomFinalStep()`
- `NewStructureTemplateStep.CREATE_STRUCTURE`

7.10 Structure 5.2 Release Notes

27th of December, 2018

Structure 5.2 adds support for Jira 8.0 and several performance improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 978) Try It: Structure Sandbox Server[265] (no installation required)

7.10.1 Version Highlights

- Adds support for Jira 8.0
- Clicking the “Open” link from a Structure gadget opens Structure Board with the same filters applied
- Manage Structure is now available from the structure selection menu
- Several performance improvements and fixes

265 https://demo-structure.almworks.com/

7.10.2 Changes in Detail

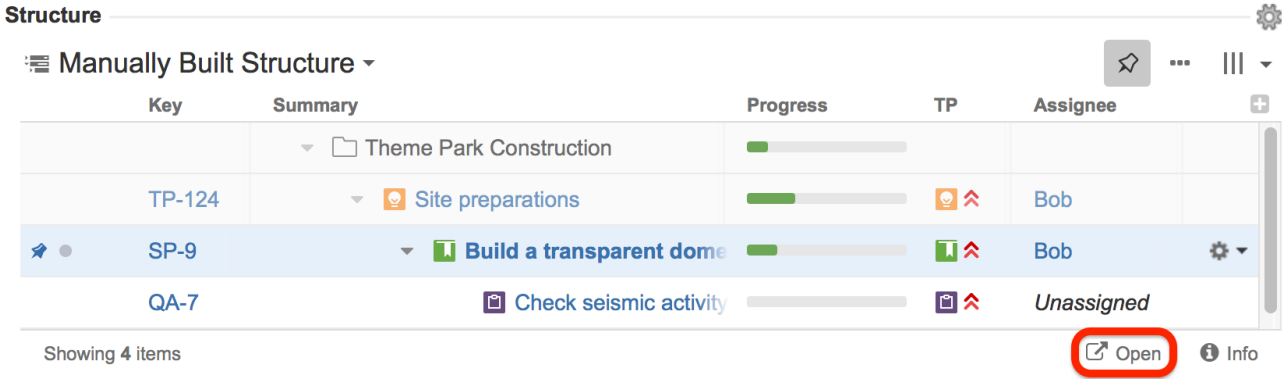
7.10.2.1 Support for Jira 8.0

Structure 5.2 is the first version to support Jira 8.0. We have thoroughly tested it against Jira 8.0 Beta and fixed all discovered incompatibilities.

> ⚠ When you upgrade to Jira 8.0, your existing index files will automatically be deleted and rebuilt using Jira's new index format. During this reindexing, all structures will appear empty and read-only, and you will see a warning message about the inconsistent index. This is normal – your structures will become available again as soon as the reindexing is complete.
>
> Additionally, background synchronization will not be possible during the reindexing. If you rely on synchronizers, you should manually resync your structures once the new index is ready.

7.10.2.2 Smoother Transition from Structure Gadget to Structure Board

When viewing a structure from anywhere other than the Structure Board, filters[266] or other transformations[267] are often applied to limit what you see to the context of that particular location. For example, when viewed from a project page, the structure is filtered to only show issues from that project. When you use the open link, those same transformations are now applied on the Structure Board.

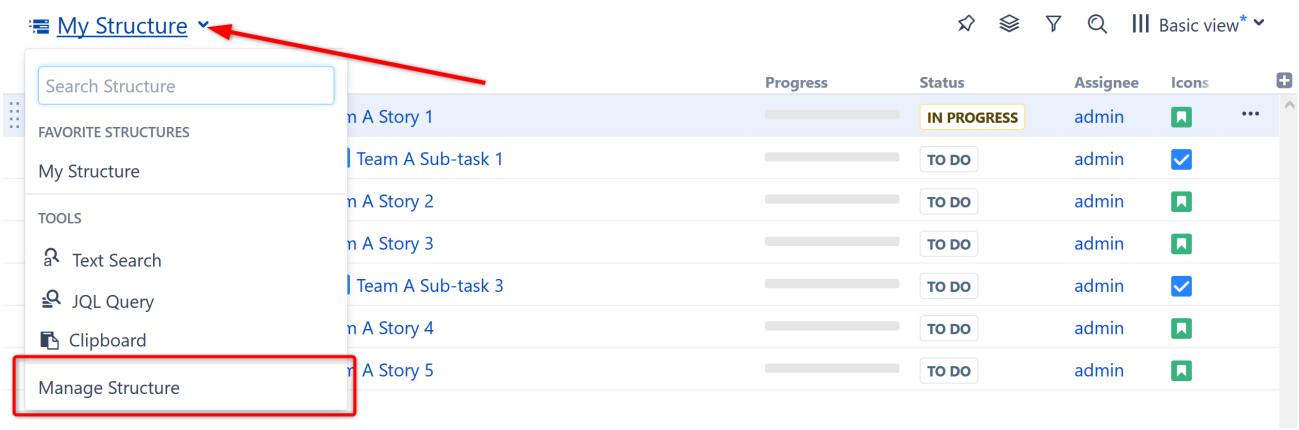


To review or remove the transformations, click the Transformations button in the panel toolbar.

7.10.2.3 Manage Structure in Structure Selection Menu

Manage Structure is now accessible from the structure selection menu.

266 https://wiki.almworks.com/display/structure2gmaster/Filter
267 https://wiki.almworks.com/display/structure2gmaster/Transformations



7.10.3 Supported Versions

Structure 5.2 and all extensions support Jira versions 7.6 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.1. Structure.Testy[268] extension, Colors[269] , Structure.Pages[270], Structure.Gantt[271] and integrations with third-party apps should continue working normally.

7.10.4 Installation and Upgrade

7.10.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 978) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on the Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please note that there are limitations to its compatibility with Structure 4.2 and higher, so a Confluence upgrade to version 6.1 or later is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.10.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

---

268 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
269 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
270 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
271 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

The upgrade procedure from versions 3.x–5.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

7.10.5 Enterprise Deployment Notes

In terms of stability and performance, this release does not bring significant changes compared to version 5.1.0, as we have mostly concentrated on compatibility, bug fixes, and minor improvements.

There are no particular special areas of interest for load testing and stress testing Structure 5.2. We advise running the same testing procedures as you've done for previous upgrades.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[272].

7.10.6 API Changes in Structure 5.2

7.10.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.6+	16.12.0

See Structure API Versions(see page 883) for full version information and downloads.

7.10.6.2 Compatible Changes in the Java API

New methods in the `Forest` interface

We have added three new methods to the `Forest` interface and its implementation, `ArrayForest`:

- `long getNextSibling(long row)`

272 mailto:support@almworks.com

- `long getNextSiblingForIndex(int index)`
- `int getNextSiblingIndex(int index)`

These methods let you find the next sibling of a row. They complement the existing methods for finding the preceding sibling.

Data Center-specific licenses

We have updated the licensing API to distinguish Jira Data Center-compatible licenses from regular server and legacy licenses. The additions include:

- `LicenseData.isDataCenterCompatible()` method checks the license is compatible with Jira Data Center;
- `LicenseData.getLicenseHosting()` method returns an instance of the new `StructureLicenseHosting` enum, which tells whether this is a Data Center license, a regular server license, or a "legacy" license.
- `StructureLicenseError.NO_DATACENTER_SUPPORT` is a new error code reported when an incompatible app license is used on a Data Center instance.

7.10.6.3 JavaScript API for adding item details support

We have added a JavaScript API that allows app developers to implement item details support for specific item types. See registerItemDetailsProvider()[273] and ItemDetailsProvider[274] for details.

There is also an API usage sample that implements item details for Jira users. It can be found on API Usage Samples(see page 965) page, see 'user-item-details'.

7.10.7 Structure 5.2.1 Release Notes

> **6th of February 2019**
>
> Structure 5.2.1 fixes a critical indexing problem that may affect users.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[275]

7.10.7.1 Patch Release

This is a patch release based on Structure 5.2.0. It fixes a critical problem that could cause a full re-index to fail because of interference from Structure-related background activity.

Upgrade is recommended for all customers using Structure 5.2.0.

---

273 https://wiki.almworks.com/display/structure2gmaster/JavaScript+API+Functions#JavaScriptAPIFunctions-window.almworks.structure.api.registerItemDetailsProvider(itemType,ProviderClass)
274 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class
275 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.10.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.10.7.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 5.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.10.7.4 Enterprise Deployment Notes

The main platform change in Structure 5.2 was the introduction of index consistency checking. To summarize, Jira 8.0 includes a newer version of Lucene, so it will delete your existing index after the upgrade and start re-indexing in the new format. Since Structure relies heavily on the Lucene index, we want to pause its operation until the re-index completes and the index becomes consistent with the database again.

To do that, Structure periodically checks the Lucene index consistency by calling a method provided by the Jira API, which performs a few database queries and index accesses. We have discussed our implementation with the Jira development team, who assured us that these periodic checks wouldn't negatively affect the performance of the Jira instance.

During a full re-index, the Lucene index is normally expected to be inconsistent, and the consistency checks should be suspended. However, due to a problem with the Jira API not reporting correctly that the index is being rebuilt, Structure was still performing these checks in the background once about every 5 seconds. In a few support cases, we have observed a strong correlation between this background activity and the re-index failing due to missing index files.

We are communicating with Jira developers to uncover the actual cause of the issue. In the meantime, to work around the problem, Structure 5.2.1 uses a different method to detect that a full re-index is in progress. Also, we have made it possible to disable index consistency checking altogether by setting a dark feature.

We advise that you try running a full re-index on a staging Jira instance before upgrading from Structure 5.1 or earlier. The usual load and stress testing can also be performed.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[276].

276 mailto:support@almworks.com

7.10.8 Structure 5.2.2 Release Notes

> **1th of March 2019**
>
> Structure 5.2.2 fixes a regression and improves compatibility with Jira 8 and Portfolio 2.24.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[277]

7.10.8.1 Patch Release

This is a patch release based on Structure 5.2.1. It fixes a regression, introduced in version 5.2.0, which could manifest in the following:

- Generators and synchronizers can produce partial or incorrect results when background re-index is in progress.
- Synchronizers don't react to issue changes made by other synchronizers.

This version also improves compatibility with Portfolio 2.24 and Jira 8 and addresses the following issues:

- Fixed: Parent Issue Grouper does not work with Portfolio 2.24.
- Fixed: Issue fields with dropdowns are not editable from Gadget on Jira 8.
- Fixed: `NoSuchMethodException` when flushing thread-local searchers.

Upgrade is recommended for all customers using Structure 5.2.x, Jira 8, or Portfolio 2.24.

7.10.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from our Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.10.8.3 Upgrade

> If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure for versions 3.x to 5.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have a proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

277 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.10.8.4 4. Enterprise Deployment Notes

Structure 5.2.2 does not introduce changes that could affect performance or would justify additional load and stress testing for Enterprise deployments.

7.11 Structure 5.1 Release Notes

25th of October, 2018

Structure 5.1 adds wiki markup, an admin interface for dark features and several performance improvements and fixes.

Download the latest version of Structure and its Extensions
(see page 978) Try It: Structure Sandbox Server[278] (no installation required)

7.11.1 Version Highlights

- Formula columns now support wiki markup
- New admin interface for enabling/disabling Structure dark features
- Inactive users are now flagged when grouping by Assignee
- Several performance improvements and fixes

Data Center customers: Due to a problem in the Atlassian Universal Plugin Manager, the previous Data Center approved version of Structure, Structure 5.0.1, was reported as incompatible. This problem does not occur with Structure 5.1.

7.11.2 Changes in Detail

7.11.2.1 Wiki Markup

Wiki markup is now supported within Formula columns(see page 197).

Using wiki markup, you can:

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

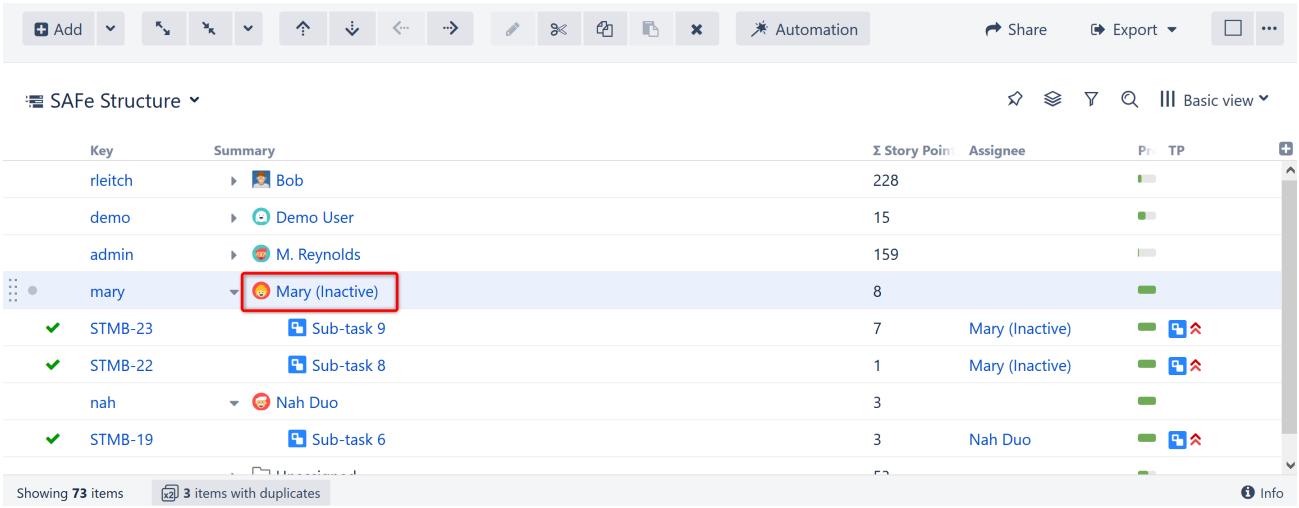
278 https://demo-structure.almworks.com/

Markup content can be included when exporting a structure to Excel or a printable page.

To learn more, see Wiki Markup in Formula Columns(see page 207).

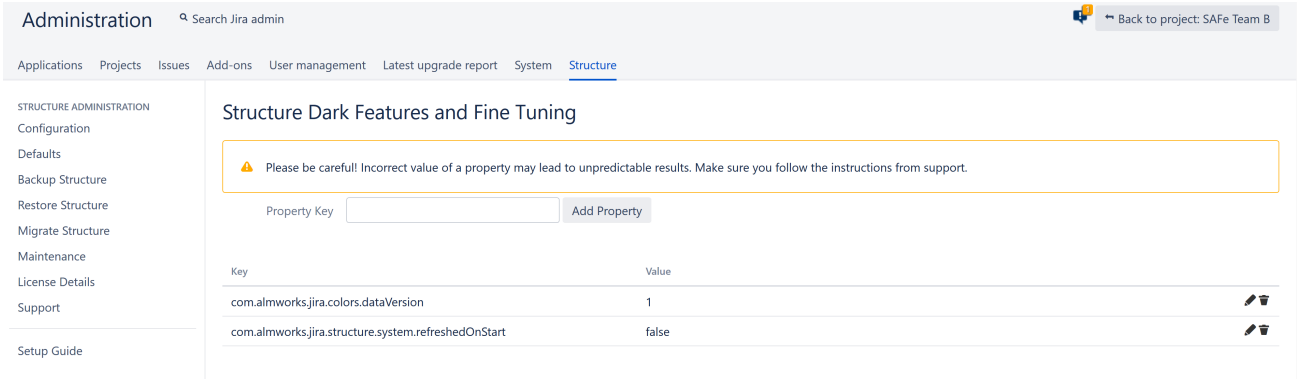
7.11.2.2 Inactive User Flag

When grouping by Assignee (either through automation or transformation), the "Inactive" identifier is now visible beside the group name:



7.11.2.3 Admin Interface for Dark Features

It is now possible to easily enable or disable dark features directly from a new Structure dark features interface.



To access the interface, you must have Jira Administration permissions and enter the interface location directly into your browser: https://YOUR_JIRA_ADDRESS/secure/admin/StructureDarkFeatures.jspa

Notes

- Properties added with our admin interface are added to Application properties
- A Structure property gets its value from Application properties; if no value is found, the property gets its value from System properties

For more information, see Advanced Configuration(see page 648).

7.11.2.4 Notable Fixes and Improvements

- Group, project and role lookup is now available when editing structure permissions
- The #distinct modifier has been added to the JOIN function in formulas, to prevent values from being added multiple times
- Shared perspectives can be viewed with anonymous access - issues that require specific permissions will not be shown in the structure
- When adding generators, the "allow changes" option can be unchecked by default
- Fixed: Exporting to Excel issue when exporting a perspective created by Structure 4.x
- Fixed: Unable to move issues under a folder when using Rank sort generator
- Fixed: Clicking column header does not sort structure when user has view permission
- Fixed: Reordering sub-tasks on the issue page does not reorder issues in the structure
- Several additional fixes and performance improvements

7.11.3 Supported Versions

Structure 5.1 and all extensions support Jira versions from 7.2 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–5.0. Structure.Testy[279] extension, Colors[280] , Structure.Pages[281], Structure.Gantt[282] and integrations with third-party apps should continue working normally.

> Structure 5.1 will be the last version to support Jira 7.2-7.5. Structure 5.2 will support Jira 7.6+.

279 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
280 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
281 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
282 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

7.11.4 Installation and Upgrade

7.11.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from the Atlassian Marketplace or our Download (see page 978) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under the "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on the Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please note that there are limitations to its compatibility with Structure 4.2 and higher, so a Confluence upgrade to version 6.1 or later is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.11.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure from versions 3.x–5.0 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

> We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 0).

7.11.5 Enterprise Deployment Notes

Structure 5.1 has a number of changes that are important for large-scale Jira Server and Jira Data Center instances.

7.11.5.1 Structure Index Monitor Field DATA CENTER

"Structure Index Monitor" is a system custom field that is automatically installed on Jira Data Center instances since Structure 4.6. It helps Structure capture events related to Lucene-based index subsystem. This is what it looks like on the Jira administrator's Custom Fields page:

This custom field does not store any data in the index, so it does not have a performance impact. Structure Index Monitor field cannot be removed, but it will disappear if Structure is uninstalled and will not affect Jira at all.

One application of this field is ensuring correct cache operation in Structure. When an issue is updated on one node, the indexes on all nodes must be updated to make sure JQL queries return the correct result everywhere. Structure caches also need to be checked on all nodes, and potentially a JQL query should be re-run to make sure that generated content is up to date. The cache update should not happen earlier than the index update, otherwise a JQL query would return an outdated result.

With Structure 5.1, we use the Structure Index Monitor field to help with the timing of cross-node updates. It eliminates a chance of a race condition that would cause Structure to show outdated content on a particular node for a long time after a relevant update has happened.

To test the rollout of this change, you can observe the behavior of the system under a constant stream of issue updates (one change per second, for example), happening on one node, while a user observes a structure with JQL-based automation on another node. At some point the stream of changes should stop, and a few seconds later the user should see the most up-to-date information in Structure.

This feature has been experimental for a while, and we're enabling it now as the default. It is still possible to turn it off and go back to the way events were propagated between nodes in Structure 5.0 and earlier, by using the dark features interface and setting `structure.delegatingItemTracker.enableReindexMonitor` to `false`.

7.11.5.2 Offloading Change Stream Writing to a Background Thread DATA CENTER EXPERIMENTAL

In the Data Center environment, Structure running on one node needs to let Structure running on other nodes know when an item (an issue or some other object) changes. This "change stream" is communicated to other nodes via the database, asynchronous caches and occasional use of a global, one-per-cluster lock.

Normally, each change is written into the database immediately when it happens – in the "event listener". That code runs in the same execution thread as the change itself, typically as a response to a user's action. We recently worked on an incident where the Jira global locking subsystem failed and there were certain issues with the database. That made writing of the change stream "hang", which, in turn, made user request threads hang, which led to Jira being unresponsive.

In Structure 5.1, we're introducing an alternative implementation of this subsystem, which would never block a user request thread. All global lock operations and writing to the database will happen in a separate, dedicated thread of execution.

This feature is currently **not** enabled by default. For now we recommend to turn it on if a similar problem has happened in the past, or if there's a problem with Jira hanging and you suspect it might be this case.

To enable this feature, set the `structure.delegatingItemTracker.enableAsyncEventsQueue` property to `true` in the dark features interface. For more information, see Advanced Configuration(see page 648).

To test the change, use the same method as described in the previous section.

7.11.5.3 Lucene and Jira Search Integration

In preparations for the upcoming Jira 8 release, we have adjusted how Structure works with Lucene index and Jira search. This change should not affect performance or the users in any way. To test that everything is okay, one could run a JQL query from Structure | Query board.

7.11.5.4 Bulk Loading Rows from the Database

One of the database tables that Structure most frequently uses is `AO_8BAD1B_ROW`. Structures contain rows, and this table stores the mapping between unique numeric "row IDs" and arbitrary "item IDs" of the objects that structures contain. This table is used very often when a structure is displayed.

With Structure 5.1, we implemented a bulk operation that lets Structure get multiple row information with one database request. That greatly speeds up loading structures that have lots of manually added issues and reduces the database load.

To test this improvement, you can try creating a temporary structure, adding several thousands of issues there (manually, copying them from a search result), and then opening that structure after an upgrade or after Structure is disabled and enabled.

7.11.5.5 Testing on Staging Environment

Apart from the specific suggestions given above, you can run the generic load and stress testing on a staging environment as advised in the previous release notes.

> Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[283].

7.11.6 API Changes in Structure 5.1

7.11.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.2+	16.11.0

See Structure API Versions(see page 883) for full version information and downloads.

7.11.6.2 Compatible Changes in the Java API

- The new `DarkFeatures` class, backed by Jira's `ApplicationProperties`, is used to obtain advanced configuration parameters where previously Structure relied on system properties. It is primarily intended to be used by Structure itself and other ALM Works products.
- `La.cast()` is a new static factory method that produces a `La` instance that safely casts objects to the given class. It complements the existing `La.instanceOf()` factory method.
- A new version of `RowTree.appendForest()` takes a `java.util.function.LongPredicate` instead of the `LongPredicate` from HPPC. The old method has been deprecated.

283 mailto:support@almworks.com

7.11.7 Structure 5.1.1 Release Notes

5th of May 2020

Structure 5.1.1 is a security patch release.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[284]

7.11.7.1 Patch Release

This is a patch release based on Structure 5.1. We have addressed a security issue that affects all Structure versions starting with 3.5. Problem was fixed in the latest (6.0) Structure.

Upgrade is required for all Jira instances running Structure version 5.1 and using Jira 7.2.x.

You should have Structure license with active maintenance (expiring not earlier than May 6th, 2020) to upgrade.

7.11.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.11.7.3 Upgrade

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes[285].

The upgrade procedure from versions 3.x–5.1 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and a proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off the "**Backup History**" option to avoid a long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

284 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
285 https://wiki.almworks.com/display/structure2gmaster/Structure+3.0+Release+Notes

7.12 Structure 5.0 Release Notes

16th of August, 2018

Structure 5.0 introduces Horizontal Scrolling, Manual Adjustments to auto-generated content, user interface updates and more.

Download the latest version of Structure and its Extensions
(see page 978)Try It: Structure Sandbox Server[286] (no installation required)

7.12.1 Version Highlights

- Manual adjustments to auto-generated content
- Horizontal scrolling
- Work Logged can now be based on user, group or role
- Flexible maintenance scheduling with crontab
- Support for “hidden” issue links
- User interface changes to match Jira 7.10

7.12.2 Changes in Detail

7.12.2.1 Manual Adjustments

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

When you use generators to build structures, you’re restricted in how you can move items within your structure, based on the generator rules. When Manual Adjustments are enabled, you can move items or create new items within your structure regardless of the generators used to create it.

Manual Adjustments are not reflected in Jira and may be affected by changes to your generators or Jira items.

Documentation: Manual Adjustments (see page 175)

Implementing Manual Adjustments required us to make changes to our backup file format. As such, previous versions of Structure will not be able to restore data from backup files created by Structure 5.0 and later. We strongly recommend that you back up your data before upgrading. For more information, see Backup Format Version Change(see page 658).

286 https://demo-structure.almworks.com/

7.12.2.2 Horizontal Scrolling

To make it easier to work with multiple columns and/or larger column widths, you can now enable horizontal scrolling within your structure. When Horizontal Scrolling is enabled, you can add columns beyond the width of your viewing pane and navigate between columns using a convenient horizontal scrollbar.

Documentation: Horizontal Scrolling(see page 333)

7.12.2.3 Work Logged by User, Group or Role

The Work Logged column has been updated so time spent on an issue can now be calculated for a specific user, group, or project role.

Documentation: Work Logged Column(see page 315)

7.12.2.4 Automatic Maintenance Scheduling with Crontab

Automatic Structure Maintenance now supports flexible scheduling with crontab.

Using standard crontab formatting, you can customize your maintenance schedule to request that tasks run only on certain days or during certain times, process multiple backups each day, specify exact times down to the minute and more.

Documentation: Automatic Structure Maintenance(see page 490)

7.12.2.5 Other Changes

Other changes include:

- Structure now supports hidden issue links
- Several user interface changes were made to match Jira 7.10 graphical changes and improve accessibility

7.12.3 Supported Versions

Structure 5.0 and all extensions support Jira versions from 7.2 or later. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.6. Structure.Testy[287] extension, Colors[288], Structure.Pages[289], Structure.Gantt[290] and integrations with third-party apps should continue working normally.

7.12.4 Installation and Upgrade

7.12.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

---

287 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
288 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
289 https://marketplace.atlassian.com/apps/1215242/structure-pages-for-jira-and-confluence?hosting=server&tab=overview
290 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

1. Download and install Structure app, either from Atlassian Marketplace or from Download(see page 978) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.12.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x–4.6 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

> We strongly recommend that you back up your data before upgrading. The introduction of manual adjustments required changes to our backup file format, which makes previous versions of Structure unable to restore data from backup files created by Structure 5.0 and later. For more information, see Backup Format Version Change(see page 658).

7.12.5 Enterprise Deployment Notes

Structure 5.0 introduces the ability to manually adjust generated content and improves the performance of the Sub-Task extender. The changes in the Automation sub-system can be particularly important for larger installations and Jira Data Center instances.

7.12.5.1 Controlling Manual Adjustments

Manual adjustments are disabled by default for new and existing structures, so they shouldn't immediately affect Structure's behavior and performance.

Structure's permission system is used to control who has access to manual adjustments:

- Enabling manual adjustments for a structure requires **Control** permission for that structure.
- Any user with the permission to create structures can create a new one and enable manual adjustments for it.
- **Edit** permission is required to add and modify manual adjustments if they are enabled.
- **Control** permission is required to remove all manual adjustments for a structure.

Documentation: Structure Permissions(see page 362), Changing Permission to Create New Structures(see page 479).

In addition to permissions, there are configurable limits on the number of manual adjustments:

- 200 adjustments per user action, controlled by the system property `structure.gfs.manualAdjustments.maxAdjustmentsPerAction`
- 2000 adjustments per structure, controlled by the system property `structure.gfs.manualAdjustments.maxAdjustmentsPerStructure`

Finally, manual adjustments can be disabled for the entire Jira instance by setting the `structure.gfs.manualAdjustments.enable` system property to `false`.

Documentation: Advanced Configuration with System Properties(see page 655)

7.12.5.2 New Database Table for Manual Adjustments

We have added a new table named `AO_8BAD1B_MANUAL_ADJUSTMENTS` to the database schema. This table stores primary data (not stored anywhere else in Jira).

The table contains one row for each structure with manual adjustments. The total amount of data stored depends on the number of adjustments and the complexity of the structure, however, a rough estimate would be on the order of 100 characters per adjustment.

The data from this table is exported along with other Structure data when you do Structure Backup in a human- and machine-readable form. The table is also exported along with all Structure data during Jira backup.

7.12.5.3 Backup Format Version Change

In Structure 5.0 we had to change the backup file format version due to the backwards-incompatible changes required by the introduction of manual adjustments. This means that previous versions of Structure will not be able to restore data from backup files created by Structure 5.0 and later. Because of this, we strongly recommend that you back up your data before upgrading.

However, if you want to downgrade to an earlier version, you can use the procedure outlined below to restore Structure data from a 5.0 backup file:

- Unpack the XML backup file from the ZIP archive created by Structure.
- Change the `version` attribute in the `<structure-backup>` element from "5.0" to "3.3".
- Delete all `<manualAdjustments>` elements from the XML.

Then you can restore directly from the modified XML file; you do not have to pack it into a ZIP archive.

7.12.5.4 Sub-Task Extender Performance Improvement

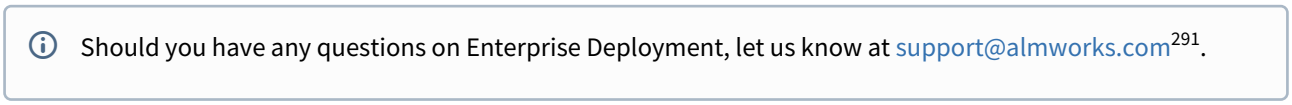
The Sub-Task extender has to check whether each issue in its scope is a sub-task or a regular issue. Previous versions of Structure relied on JQL to perform issue type checks, and on large Jira instances with hundreds of issue types these JQL queries caused significant performance impact. In Structure 5.0 we have changed the Sub-Task extender to read issue types from the Lucene index directly, instead of using JQL. We have also improved caching, so sub-task extenders are now better able to reuse the results of issue type checks done by other sub-task extenders.

7.12.5.5 Testing on Staging Environment

The introduction of manual adjustments required extensive changes throughout the Automation sub-system, so we advise you to perform load and stress testing in a staging environment before you upgrade. You can try the following scenarios:

- If you have large structures (10,000 issues or more) built by generators, especially sub-task extenders, try opening those from a number of client computers using different user accounts.
- Make some changes to Jira issues that would cause these structures to regenerate – change field values, add or remove links, change sub-task parents.
- Make changes in the structures that would be processed by generators – move issues to different groups, move linked issues and sub-tasks to different parent issues.
- Create a new structure and add manual adjustments to it. For example, you can rearrange a flat list of issues added by a JQL inserter into a hierarchy, or move the issues into several manually added folders.

Watch the log files for errors and warnings while running these experiments.



7.12.6 API Changes in Structure 5.0

7.12.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, except those in `@Internal` interfaces.

Jira Version	New API Version
7.2+	16.10.0

See Structure API Versions(see page 883) for full version information and downloads.

7.12.6.2 Compatible Changes in the Java API

Horizontal scrolling

We have added a new parameter named "column display mode" to the view specification, with three new getters and setters:

- `ViewSpecification.getColumnDisplayMode()`
- `ViewSpecification.Builder.getColumnDisplayMode()`
- `ViewSpecification.Builder.setColumnDisplayMode(int)`

Supported values are stored as `int` constants in the new class `ColumnDisplayMode`:

- `AUTO_FIT` means that the columns fill the entire width of the panel, growing or shrinking as necessary;
- `SCROLLABLE` means that the columns always have their manual or preferred widths, and a horizontal scroll bar appears if needed.

The default column display mode is `AUTO_FIT`.

291 mailto:support@almworks.com

New method in the `Forest` interface

The new method `copySubforestAtIndex(int)` returns a new forest that contains all descendants of the row at the given index. It is a counterpart to the existing method `copySubforest(long)`, and it allows you to save an index lookup when you already have the index. The method is implemented in `ArrayForest`.

New row semantics

We have added two new codes to the `CoreSemantics` interface:

- `INSERTED` marks rows added by inserter generators;
- `EXTENSION` marks rows added by extenders.

Please note that row semantics is still a work in progress, and `CoreSemantics` is marked `@Internal`.

Miscellaneous fixes

A few minor changes in `AbstractAggregateLoader`, `AbstractDistinctAggregateLoader`, and `ReductionStrategy` are meant to prevent `NullPointerException` in certain circumstances.

We have also added reentrancy protection to the item lookup code in `StructureException`. Please note that item lookup is disabled by default since version 16.7.0 (Structure 4.4).

7.12.6.3 Incompatible changes in `@Internal` interfaces

The method `ForestAccessCache.removeOutOfScopeIssues()` was moved to a separate interface, `ProjectScopeCache`. We have also added a new internal interface named `SubTaskIssueCache`. Neither of the three interfaces is intended to be used by third-party developers at this point.

7.13 Structure 4.6 Release Notes

27th of March, 2018

Structure 4.6 introduces Automation Timeout safeguard and inline editing of the Status field.

Download the latest version of Structure and its Extensions
(see page 978)Try It: Structure Sandbox Server[292] (no installation required)

7.13.1 Version Highlights

- Automation Timeout safeguards Jira against excessive load that may result from misconfiguration of Structure generators
- Inline editing of the Status field enables users to change an issue's status with a double-click on the Structure grid

292 https://demo-structure.almworks.com/

7.13.2 Changes in Detail

7.13.2.1 Automation Timeout

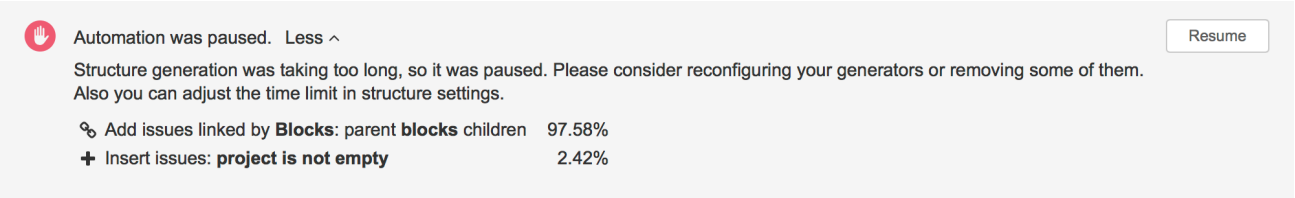
The Structure Automation(see page 122) feature is very powerful, and very flexible. It gives the user the building blocks – generators – for constructing dynamic structures. As with any powerful and flexible feature, it's possible to unintentionally create an unanticipated load on the Jira server. The Automation Timeout is a safeguard that protects Jira from unintended server loads.

Some generators already have a considerable level of protection. For example, JQL Inserter offers an option to limit the maximum number of issues it adds to a structure. Extenders have a maximum number of levels to expand a hierarchy. Automation Timeout adds universal protection on top of that by limiting the amount of time Structure app can spend generating a particular structure.

Here's how it works.

When Structure app detects that a structure is being generated for an unreasonably long time (by default, 30 seconds), it **pauses Automation** for that structure. The Structure panel will show the "**Automation was paused**" message and the structure itself will miss the generated content. If the structure contains only dynamic elements, it will appear empty.

If the user clicks "More" link, they can inspect which generators took most of the time.



Any user who can view the structure can resume Automation by clicking **Resume** button. This will cause Structure to try and generate this structure again.

> Note that sometimes structure generation may time out because of general slowness of Jira caused by other factors. It is ok to click Resume and try again in that case, without changing the generators.

Documentation: Paused Automation(see page 660)

7.13.2.2 Inline Status Editing

It is now possible to edit an issue's Status inline by double-clicking on its current status in the Status column.

You can then select a new status from a list. The list will show only the statuses that could be the issue's next status, according to the workflow. If a particular transition includes a screen with additional fields, this transition cannot be executed through inline editing and will require the use of Jira Actions(see page 121).

Documentation: Editing Issues(see page 114)

7.13.2.3 Notable Fixes and Improvements

- Fixed: When exporting a structure to a Printable page in Automation Editing Mode, some rows are not shown
- Fixed: When exporting a structure to Excel, large texts are truncated to 16382 symbols

- Fixed: Issues can be added to archived Fix Versions
- Fixed: Progress based on Time Tracking takes folders and pages into calculation
- Fixed: Folders are not shown in the structure on a product page
- Fixed: Duplicates filter is not included in a shared perspective

7.13.3 Supported Versions

Structure 4.6 and all extensions support Jira versions from 7.2 to 7.8. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.5. Structure.Testy[293] extension, Colors[294], Structure.Gantt[295] and integrations with third-party apps should continue working normally.

7.13.4 Installation and Upgrade

7.13.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure app, either from Atlassian Marketplace or from Download(see page 978) page.
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for Structure.Pages and Structure Helper apps, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.13.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x–4.5 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

293 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
294 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview
295 https://marketplace.atlassian.com/plugins/com.almworks.structure.gantt/server/overview

7.13.5 Enterprise Deployment Notes

Structure 4.6 has a number of changes that are particularly important for large installations and Jira Data Center instances.

7.13.5.1 Automation Timeout

One of the problems we've seen in the past is that one or more improperly configured structures would take a lot of time to generate, consuming system resources. In extreme and rare occasions, system load could make entire Jira instance unresponsive. To prevent that from happening, we're introducing a time limit for structure generation. If a structure takes longer time to get generated, it will be marked as "paused", and all generators in it will be temporarily disabled.

The default timeout is 30 seconds. If some of your structures currently take a long time to generate, they might get paused after the upgrade.

Therefore, we advise you to carfeully check your all of your bigger and mission-critical structures in a staging environment before upgrading and adjust the timeouts as necessary. The structures with paused automation will be marked as such on the Manage Structures page.

> Note that someone needs to actually open a structure for Automation to run the generation process. A structure that is not being looked at does not consume any system resources.

The timeouts can be adjusted for each structure using "Configure" link on the Manage Structure page. The global default can be adjusted in the **Administration | Structure | Defaults** menu section.

7.13.5.2 Synchronizers on Jira Data Center

Structure uses only one cluster node to run synchronizers. We've seen some problems with the mechanism used to select a cluster node for running synchronization and hand off its work to a different node in case the original node is stopped or loses connectivity. These problems could cause all synchronizers in the system to stop working.

In this version we're introducing a simpler and more reliable mechanism which uses conditional database updates and Jira's built-in node aliveness checks. A new database table, AO_8BAD1B_ATOMIC_FLAG, is introduced to support the new solution. In this version, it is expected to contain only a single row that shows which node is currently responsible for running synchronizers.

You can use the following steps to test syncronization node selection and hand-off:

1. Start Jira (Data Center), install Structure 4.6 and check application logs for messages from *ClusterExclusiveWorkNodeFlag* to determine which node is currently responsible for synchronization.
2. Shut down or disconnect that node and wait for up to 6 minutes, which is the maximum reaction time.
3. Check application logs for messages from *ClusterExclusiveWorkNodeFlag* to see that synchronization was taken over by another live node.

7.13.5.3 Lucene Index Monitoring

Structure relies on the Lucene index to check users' access to issues, or quickly read a single value for a large number of issues. The index may be temporarily inconsistent during full or project reindexing, index replication or recovery. Structure 4.2 introduced re-index monitoring as a part of a dark feature used to work around a race condition in Jira Data Center. We've updated this mechanism in Structure 4.6 to also handle index replication and

automatic recovery, and now we're enabling it for all instances, so that Structure is able to recover when it detects massive index changes.

After the upgrade, a new, undeletable system custom field called "Structure Index Monitor" should appear. This field is used only to track the "reindex" events. It does not contain any data and will not have any effect on issues. If Structure is uninstalled, this custom field disappears.

7.13.5.4 Old Row Manager Implementation Removed

Structure 3.4 introduced a new implementation of one of its core components used to store temporary rows in the generated structures. The old implementation remained in the code with the ability to switch to it by setting a system property. In over a year since that release the new implementation has proven to be reliable, and we never advised our customers to switch to the old one. In Structure 4.6 we remove the old implementation and its dependencies completely.

If you see folders named *rows0*, *rows1*, etc., under *JIRA_HOME/structure* folder, they can be safely deleted – they were a part of the older implementation.

7.13.5.5 Testing on Staging Environment

There are no particular special areas of interest for load testing and stress testing Structure 4.6. We advise running the same testing procedures as you've done for previous upgrades.

> Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[296].

7.13.6 API Changes in Structure 4.6

7.13.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.2+	16.9.0

See Structure API Versions(see page 883) for full version information and downloads.

7.13.6.2 Compatible Changes in the Java API

Automation timeout

We have added two new methods to notify Structure API users that the generation of a structure was stopped due to a timeout:

296 mailto:support@almworks.com

- `ForestSource.getHealth()`
- `VersionedForestUpdate.getHealth()`

These methods return a `ForestSourceHealthStatus` object, and its `isStopped()` method returns true if generation was stopped.

`StructureErrors.AUTOMATION_FAILED` is a new error which may be thrown when automation is paused and there is no way to return a health status object.

We have also added a new method named `isStopped()` to the `GenerationContext` interface. If you implement your own generators, we advise you to check this method before potentially long-running operations and inside loops, and return control back to Structure if it returns true.

Convenience methods in `StructurePropertyService`

We have added two new methods to the StructurePropertyService interface:

- `getLong(long, String)` returns the property value as a `long`;
- `setValue(long, String, long)` sets the property to the given `long` value.

`@Internal` components and classes

Two interfaces have been moved from Structure core to a new API package, `com.almworks.jira.structure.api.statistics`, for consumption by other ALM Works products. They are not intended to be used by third-party developers at this point.

7.13.7 Structure 4.6.1 Release Notes

27th of April 2018

Structure 4.6.1 fixes a compatibility issue with Jira Service Desk and a couple of compatibility issues with other apps.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[297]

7.13.7.1 Patch Release

This is a patch release based on Structure 4.6.

Upgrade is recommended for all customers who use Structure with Jira Service Desk.

We have addressed the following issues:

- Fixed: Jira Service Desk users without access to Structure can't create and edit queues.
- Fixed: Epic Sum Up App time field in a structure view updates unexpectedly.
- Fixed: Xray App execution status is not rendered correctly in Structure column.

297 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

⚠ You should have Structure license with active maintenance (expiring not earlier than April 27th, 2018) to upgrade.

7.13.7.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.13.7.3 Upgrade

⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.13.7.4 Enterprise Deployment Notes

Structure 4.6.1 does not introduce changes that could affect performance or would justify additional testing for Enterprise deployments.

7.13.8 Structure 4.6.3 Release Notes

ⓘ **11th of May 2018**

Structure 4.6.3 fixes a problem in Performance Audit Log diagnostics system and has several performance improvements.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[298]

7.13.8.1 Patch Release

This is a patch release based on Structure 4.6.1. (There was no 4.6.2 release.)

298 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

We have fixed a problem that caused collection of Performance Audit Log to fail when Jira had a saved filter with an empty JQL, a JQL with only an "order by" clause, or an invalid JQL. This problem could prevent collection of valuable diagnostic data for ALM Works support team.

We also added five various performance improvements that are important for large instances (Server and Data Center), especially when there are frequent issue changes (for example, one issue per second).

Upgrade is recommended if you have encountered the problem with Performance Audit Log or if you have a large Jira instance.

⚠ You need a Structure license with active maintenance (expiring not earlier than May 11th, 2018) to upgrade.

7.13.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from the Download(see page 978) page.
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.13.8.3 Upgrade

⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.13.8.4 Enterprise Deployment Notes

There is a number of small changes in Structure 4.6.3 that are not visible to the user (except for improved performance).

The following a is a list of affected components with suggestions of how they could be tested on a staging server.

1. Attribute loading subsystem: it is used when values are loaded into the Structure grid. We removed additional checks for concurrent changes that happened during the loading of attributes. This will speed up the attribute loading, and on a busy system it will drastically increase the responsiveness of the application. To test the subsystem, open different structures and make changes to the issues. Check how totals and formulas are being recalculated. Try to load-test the staging instance by having a constant flow of issues changes while a structure with an inserter and a generator are being used.
2. Sub-task extender, Agile Rank sorter, Agile synchronizer, Sub-task synchronizer: the performance of these components was improved by changing the way issues are checked for being sub-tasks. The performance improvement will have bigger effect on systems with hundreds of issue types.

3. Automation subsystem: eliminated some inefficiencies during generator execution. To test, check the most frequently used automated structures. Try changing values in Jira and see the generated structure update. If you are using issue security, verify a limited user's access to issues. Check how transformations and quick transformations work.

Additionally, we improved the user interface of Performance Audit Log.

7.13.9 Structure 4.6.5 Release Notes

> **13th of June 2018**
>
> Structure 4.6.5 adds compatibility with Jira 7.10 and several performance improvements.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[299]

7.13.9.1 1. Patch Release

This is a patch release based on Structure 4.6.3. (There was no 4.6.4 release.)

Starting with this version, Structure is compatible with Jira 7.10.

There are two known issues:

- We are still working on making Structure's user interface match the new look and feel of Jira 7.10. The app is perfectly usable, but you can still see some older-style controls. We are planning to complete the redesign in the next feature version.
- Archiving a project, which is now possible with Jira 7.10 Data Center edition, will cause all archived issues to disappear from all structures. See Archived Projects and Structure(see page 496) for details.

This upgrade is recommended for all large instances. Depending on Structure usage, this release may greatly improve Structure performance – specifically, reducing the time it takes to load structures with automation based on Portfolio or Jira links. The improvement is especially noticeable on larger instances with a large number of projects.

This upgrade is required for instances running Jira 7.10.

> You need a Structure license with active maintenance (expiring no earlier than June 13th, 2018) to upgrade.

2. Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install the Structure add-on, either from Atlassian Marketplace or from the Download(see page 978) page.
2. When the Add-on Manager reports a successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under the "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

---

299 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

3. Upgrade

If you are upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

The upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up Jira data, using **Administration | System | Backup System**. Starting from version 3.0.0, Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

4. Enterprise Deployment Notes

Structure 4.6.5 has a few performance improvements especially important for large Jira instances with hundreds and thousands of projects.

When Structure is enabled only for certain Jira projects, generators have to make sure that they are adding issues only from those projects. Structure relied on JQL to perform project checks, producing JQL queries with clauses like: `project IN (<list of selected project IDs>)`. On large Jira instances with thousands of projects and hundreds of those projects using Structure, these JQL queries cause significant performance impact, especially when run very often by Portfolio, Link, and Agile extenders.

In Structure 4.6.5, we have changed the project checking code to read project IDs from Lucene index directly, instead of using JQL. We have also improved caching, so generators are now better able to reuse the results of project checks done by other generators. We have seen significant performance improvements with these changes in the scenario described above.

The changes affect Portfolio, Link, and Agile extenders and, to a lesser extent, JQL, Agile, and Text Query inserters. If you have large structures (about 10,000 issues or more) built using these generators, we advise you to perform load and stress testing on a staging environment before upgrading, even if your Jira instance doesn't have a lot of projects.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[300].

7.14 Structure 4.5 Release Notes

28th of December, 2017

This version introduces duplicate items highlighting, which makes it much easier to find, review and delete duplicate issues in a structure. This version also includes several performance improvements and bug fixes.

Download the latest Structure and Extensions
(see page 978) Structure Demo Server[301]

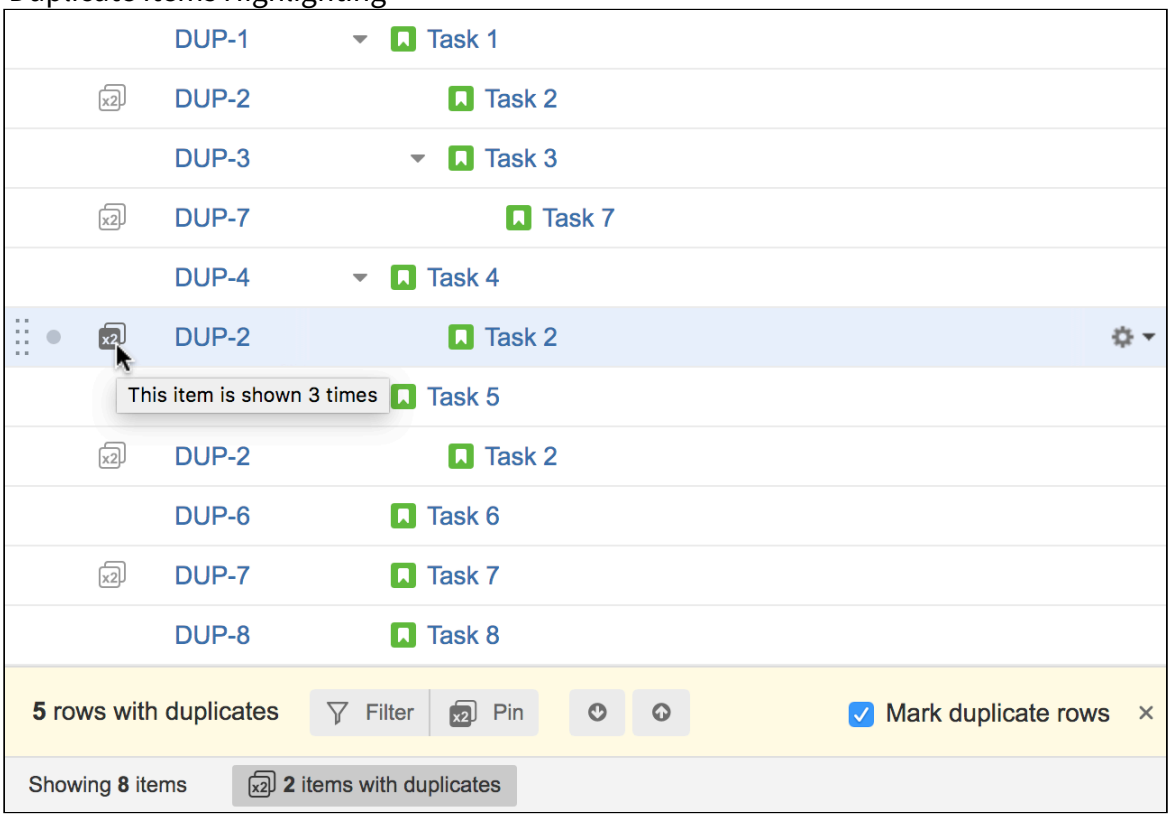
300 mailto:support@almworks.com
301 https://demo-structure.almworks.com/

7.14.1 Version Highlights

- Duplicate items highlighting
- Several performance improvements and fixes

7.14.2 Changes in Detail

7.14.2.1 Duplicate Items Highlighting



Issues can appear in structure more than once both when added manually or automatically. You can now quickly find, highlight and switch between such issues in a structure. This makes working with duplicates much simpler. For example, if necessary, you can delete manually added duplicates or adjust automation settings to make sure automation only adds one instance of an issue.

Documentation: Identifying Duplicate Items(see page 194)

7.14.2.2 Notable Fixes and Improvements

- When the issue details panel is closed, previous layout is restored
- Fixed: Performance Audit Log throws an error if Jira saved filters are used in generators
- Fixed: Level option is not saved when quick transformations are created
- Fixed: Unresolved issues from closed sprints do not show under Backlog when grouped by sprint
- Fixed: Browser search prompt changes drag and drop behavior in Structure to Copy when Move is expected
- Fixed: Show Results button does not appear when backing up Structure in a Jira Data Center instance

- Fixed: Text Search inserter moves modified issues to the bottom
- Various performance improvements

7.14.3 Supported Versions

Structure 4.5 and all extensions support Jira versions from 7.2 to 7.6. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.4. Structure.Testy[302] extension, Colors[303] plugin and integrations with third-party add-ons should work.

7.14.4 Installation and Upgrade

7.14.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your Jira version!
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for .Pages and Helper add-on, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.14.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x–4.4 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.14.5 Enterprise Deployment Notes

Structure 4.5 has a number of changes that are particularly important for large installations and Jira Data Center instances.

---

302 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
303 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.14.5.1 Extender Performance Optimizations

Big and deep structures built with multiple extenders have been a source of performance problems for some of our customers. Structure 4.5 contains numerous optimizations aimed at reducing both the generation time and memory consumption for such structures.

The changes to the extender implementations and the automation engine are quite substantial, so if you rely on large structures (10,000 issues or more) built with extenders, we advise you to perform load and stress testing on a staging environment before upgrading.

7.14.5.2 Other Changes

A fix in the Sprint grouper required it to load more data from Lucene. We do not expect this to be a problem, but if you rely on the Sprint grouper, it might make sense to test it before upgrading.

We have also optimized the attribute sub-system to speed up aggregate calculations on large, deep structures.

7.14.5.3 Testing on Staging Environment

Given the changes described above, you can test the following scenarios:

- Create a large, deep structure built with one or more extenders, for example, 10,000 issues or more, organized into 5 or more levels of hierarchy. Change one or more issues in the structure — add or remove issue links, change sub-task parents, etc., depending on the extenders you are using.
- While the structure described above is open, add one or more aggregate columns to your view (e.g. “Count Leaves” or a sum of a numeric field).
- Create a large structure (10,000 issues or more) consisting of a JQL inserter and a Sprint grouper (make sure most of the issues in the structure belong to one or more sprints). Move one or more issues to a different sprint.

Watch the log files for errors and warnings while running these experiments.

The usual load and stress testing can also be applied.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[304].

7.14.6 API Changes in Structure 4.5

7.14.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.2+	16.8.0

304 mailto:support@almworks.com

See Structure API Versions(see page 883) for full version information and downloads.

7.14.6.2 Compatible Changes in the Java API

Generic items

Generic items are simple items managed by Structure. They are much like Structure's folders, but they can also contain an icon and a description. Generic items are meant to be used by other ALM Works products and third-party apps that integrate with Structure. Server-side APIs for generic items were introduced in Structure 4.4, and with the current release we are shipping client-side and UI code, making it possible to use generic item types in third-party apps.

Documentation: Declaring a New Generic Item Type(see page 880)

`GenerationContext` changes

We have added a few methods to the `GenerationContext` interface.

- `itemsUpdate(DataVersion)` returns an item version update since the given version. It can be used to invalidate item value caches kept in the generated forest.
- `putTempObject()` and `getTempObject()` allow the developer to associate arbitrary data with the current generation context. Unlike `putObject()`, `putTempObject()` does not keep the passed objects in memory after the current generation task completes.

`ArrayForest` changes

A new method named `addForestMutuallyExclusive()` was added to `ArrayForest`. It is a faster alternative to `addForest()` and `mergeForest()` that can be used when the added forest is guaranteed to be mutually exclusive with the forest it's added to.

We have also optimized the implementation of another similar method, `replaceSubtreesMutuallyExclusive()`, which now performs fewer memory allocations when called frequently.

Miscellaneous

- Added `StructureUtil.nonBlank()`, which is used throughout the code to protect against null, empty and blank strings.
- Added a copying constructor to `ProcessDisplayParameters`.

7.14.7 Structure 4.5.1 Release Notes

19th of January 2018

Structure 4.5.1 adds compatibility with Portfolio version 2.11 and fixes other small issues.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[305]

305 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.14.7.1 Patch Release

This is a patch release based on Structure 4.5 and providing compatibility with Portfolio for JIRA version 2.11.

Upgrade is recommended for all customers who use Structure with Portfolio 2.11.

> ⚠ You should have Structure license with active maintenance (expiring not earlier than January 19th, 2018) to upgrade.

7.14.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.14.7.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
> - If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.14.7.4 Enterprise Deployment Notes

Structure 4.5.1 does not introduce changes that could affect performance or would justify additional testing for Enterprise deployments.

7.15 Structure 4.4 Release Notes

29th of November, 2017

Now you can quickly change issue status simply by dragging it from one Status group folder into another. This version also contains several bug-fixes and some improvements for our performance diagnostics tool.

Download the latest Structure and Extensions
(see page 978) Structure Demo Server[306]

7.15.1 Version Highlights

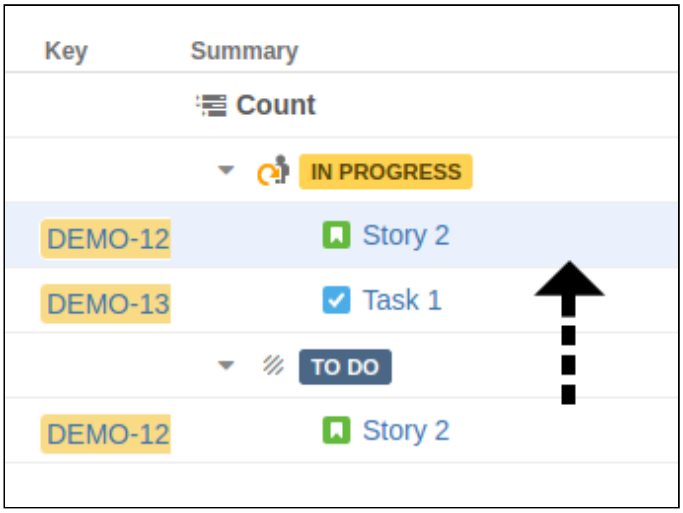
- Changing issue status using drag and drop.
- Support for Jira 7.6.
- Issue count is now displayed for the selected panel in double grid layout.
- Performance Audit Log includes more information.

7.15.2 Changes in Detail

7.15.2.1 Changing issue status in a structure

It is now possible to transition issues between statuses within a structure. Transitions that are allowed and do not have special workflow requirements (like required comments) can be done by simply moving issues between status groups.

Documentation: Groups(see page 141)



7.15.2.2 Totals in double grid

In the double-grid layout, the number of items is now displayed for the panel that is currently in focus, as opposed to only left panel.

306 https://demo-structure.almworks.com/

7.15.2.3 Changes to Performance Audit Log

Information about saved filters and filters used in generators or synchronizers is now available in Performance Audit Log.

It is now possible to save Performance Audit Log without scrolling to the bottom of the page: Save to File button is duplicated at the top.

7.15.2.4 Notable fixes and improvements

- Grouping by Customer Request Type is now possible with Service Desk 3.9.0.
- Fixed: Formula format would not work when *Sum over sub-items* checkbox was active.
- Fixed: Due In column would not show *Overdue* when past Due Date.
- Fixed: Work logged would not be displayed when filtered by users with usernames not in lower case.

7.15.3 Supported Versions

Structure 4.4 and all extensions support Jira versions from 7.2 to 7.6. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.3. Structure.Testy[307] extension, Colors[308] plugin, integrations with our partner add-ons should work with the new version.

7.15.4 Installation and Upgrade

7.15.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your Jira version!
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for .Pages and Helper add-on, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.15.4.2 Upgrading Structure

⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

307 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
308 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

Upgrade procedure from versions 3.x–4.3 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.15.5 Enterprise Deployment Notes

Structure 4.4 does not have a lot of core changes compared to Structure 4.3.

7.15.5.1 Index Monitoring Improvements

In Structure 4.2 we introduced an experimental "dark feature" for Jira Data Center to work around a race between Structure's issue change notifications and Jira's own inter-node messages which cause changed issues to be re-indexed on all nodes. You can refer to Structure 4.2 release notes, section 5.1(see page 687) for more information and detailed instructions.

Structure 4.4 improves this experimental event distribution subsystem to better handle full and single-project re-indexing operations. The changes are quite significant, so if you rely on this feature, it makes sense to test the new version on a staging environment before upgrading. If you have any questions, please contact support@almworks.com[309].

7.15.5.2 New Table for Generic Items

Generic items are simple items managed by Structure. They are much like Structure's folders, but they can also contain an icon and a description. Generic items are meant to be used by other ALM Works products and third-party apps that integrate with Structure. Only the server-side APIs and components are released with Structure 4.4, so it is not possible for users to create generic items at this point.

With the introduction of generic items we have added `AO_8BAD1B_GENERIC_ITEM` table to the database schema. This table stores primary data (not stored anywhere else in Jira).

The data from this table is exported along other structure data when you do Structure Backup in a human and machine readable form. The table is also exported along with all Structure data during Jira backup.

We don't expect considerable database load on this table.

7.15.5.3 Testing on Staging Environment

There are no particular special areas of interest for load testing and stress testing Structure 4.4. We advise running the same testing procedures as you've done for previous upgrades.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[310].

309 mailto:support@almworks.com
310 mailto:support@almworks.com

7.15.6 API Changes in Structure 4.4

7.15.6.1 Minor Java API Release

There are a few API additions coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

Jira Version	New API Version
7.2+	16.7.0

See Structure API Versions(see page 883) for full version information and downloads.

7.15.6.2 Compatible Changes in the Java API

Generic items

Generic items are simple items managed by Structure. They are much like Structure's folders, but they can also contain an icon and a description. Generic items are meant to be used by other ALM Works products and third-party apps that integrate with Structure. We are still working on this feature, and only the server-side APIs are released with Structure 4.4. Javadocs and developer documentation for generic items will be released in a future version.

`StructureException` changes

`StructureException` is thrown by many Structure API calls. In previous versions, if an exception was associated with an item or a row, it would add the item description to its message. This has proven problematic when fetching the item itself would fail, so in version 16.7.0 this feature has been disabled by default. You can enable it by setting the "structure.exception.enable.lookup" system property to "true", but this is not recommended on a production system.

`@Internal` components and classes

Several classes have been moved from Structure core to the API for consumption by other ALM Works products. Two new packages have been added – `com.almworks.jira.structure.api.effect` and `com.almworks.jira.structure.api.rest.effect`. They are not intended to be used by third-party developers at this point.

We have also added a new internal utility method, `StructureUtil.isDevMode()`.

7.16 Structure 4.3 Release Notes

20th of October, 2017

Structure 4.3 adds more aggregate functions for the Formulas columns, introduces support for the notes column in formulas and introduces filtering by Sprint status. This version also contains a minor security fix for those who use Notes column, other improvements and fixes.

Download the latest Structure and Extensions
(see page 978) Structure Demo Server[311]

7.16.1 Version Highlights

- New aggregate functions.
- Modifiers of aggregate functions.
- Show and filter by Sprint statuses (future, closed, current).
- Notes column can now be used in formulas.

7.16.2 Changes in Detail

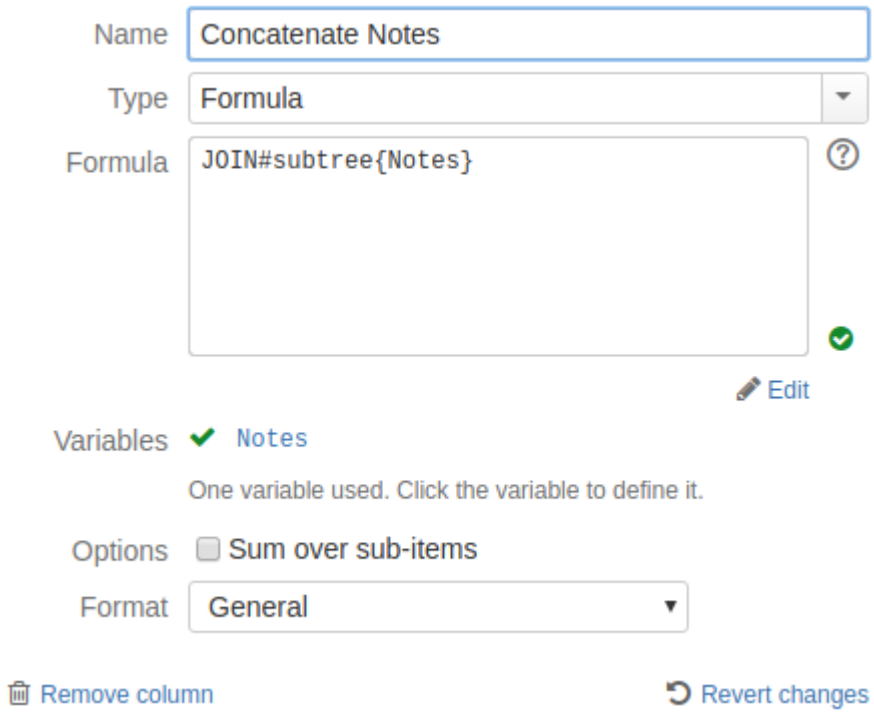
7.16.2.1 New Aggregate functions and modifiers

Further expanding on **aggregate functions** functionality, new functions are introduced: `COUNT{}`, `AVG{}`, `MAX{}`, `MIN{}`, `JOIN{}`, `PARENT{}`

Aggregate functions can also be modified to aggregate over different groups. Modifiers include but are not limited to *#all, #children, #subtree, #leaves*

Example of usage of the new syntax would be `JOIN#leaves{X}`, which concatenates contents of field X for all the leaves of the current item's sub-branch.

Documentation: Expr Language(see page 221), Aggregate Function Reference(see page 246)



311 https://demo-structure.almworks.com/

7.16.2.2 Using notes column in formulas

Notes column can now be used in formula column calculations as a variable.

It is important to mention, that notes column values are defined per structure, thus calculations including notes can return different results for the same issue in different structures.

7.16.2.3 Filtering by sprint status

Using filtering generator it is now possible to display only issues assigned to Future, Closed or Current sprints or combination thereof.

Documentation: Transformations page[312] and Filter page[313].

7.16.2.4 Security patch

We have addressed a minor security issue, affecting all Structure versions starting with 4.1 for those who use Notes column.

Details of the issue are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

If you cannot upgrade to Structure 4.3 because you are using Jira 7.1, please make sure to install Structure 4.1.1(see page 696).

7.16.2.5 Notable fixes and improvements

- It is now possible to group by Service Desk "Request type" field
- Fixed: Sub-task extender occasionally would not remove no longer matching sub-task from structure
- Fixed: Issues would not appear under Assignee when added from Search&Add panel
- Fixed: On macOS columns would overlap **+** (Add column) button
- Fixed: When clicking on the value of label-type custom field, transformation would be created around "labels" field instead

This version also includes performance improvements for static and generator-based structures.

7.16.3 Supported Versions

Structure 4.3 and all extensions support Jira versions from 7.2 to 7.5. All editions of Jira (Jira Core, Jira Software, Jira Service Desk) are supported. Jira Data Center is supported.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.2. Structure.Testy[314] extension, Colors[315] plugin, integrations with our partner add-ons should work with the new version.

312 https://wiki.almworks.com/display/structure2gmaster/Transformations
313 https://wiki.almworks.com/display/structure2gmaster/Filter
314 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
315 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.16.4 Installation and Upgrade

7.16.4.1 Installing Structure

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your Jira version!
2. When Add-on Manager reports the successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on Jira and on Confluence side. If your Confluence version is not compatible with Structure Helper 1.3, you should stay with version 1.2 for .Pages and Helper add-on, but please, note that there are limitations to its compatibility with Structure 4.2 and higher, so Confluence upgrade to version 6.1 or better is recommended.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.16.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x–4.2 is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large Jira, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.16.5 Enterprise Deployment Notes

Structure 4.3 has a few changes in the core parts that could potentially affect performance and stability of the system.

7.16.5.1 Formulas and Attribute Subsystem

The Formula column is an important Structure feature that allows users to define and calculate arbitrary metrics for structures of issues. New aggregate functions and formula features in Structure 4.3 required significant changes to the formula engine and the attributes subsystem. If you use formulas and aggregate columns with big structures (10,000 issues or more), it makes sense to check their performance on a staging system before upgrading.

The expansion of formula feature may invite users to create new automated structures and running a formula with aggregation over those structures. On large Jira instances this may lead to increased server load, as formulas with aggregation may need to scan a lot of issues to calculate a total. It's a good idea to advise the users to calculate metrics on such large structures only if there's a real need. In most cases, structures can be made smaller. When in doubt, feel free to reach out to support@almworks.com[316].

316 mailto:support@almworks.com

7.16.5.2 Time Tracking Section Changes

Structure automatically sums up time tracking information and displays aggregate values in the time tracking section on the issue page. This feature has caused significant server load in cases when multiple users were actively working with issues from multiple big structures. In order to improve time tracking section performance in Structure 4.3 we have relaxed its security checks — now the time tracking section sums up the data for all issues in the current structure, including those that the current user cannot see because of issue security restrictions. This also means that all users will now see the same values in the time tracking section. If you rely on the Structure-adjusted time tracking section, please carefully assess these changes before upgrading. If you have any questions, please contact support@almworks.com[317].

7.16.5.3 Automation Engine Improvements

Structure 4.3 contains certain fixes and performance improvements in the automation engine. Large updates of large static structures, like moving hundreds of issues at once in a structure of tens of thousands (with no automation), should be processed much faster. Another change improves the performance of level-limited extenders, which should reduce server load during large automated structure regenerations. Because the automation engine is one of the most complex parts of Structure's core functionality, it makes sense to perform load and stress testing before upgrading.

7.16.5.4 Testing on Staging Environment

You can try load testing and stress testing Structure 4.3 and Jira on a staging environment before upgrading.

It makes sense to test the following parts:

- Open large structures and add multiple predefined columns from Calculated section. Do that from a number of client computers using different user accounts.
- Loading Jira's issue page for many issues by many users at once. (Use issues from projects that are enabled for Structure.)

The usual load and stress testing can also be applied.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[318].

7.16.6 API Changes in Structure 4.3

7.16.6.1 Minor Java API Release

There are a few moderate changes coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

317 mailto:support@almworks.com
318 mailto:support@almworks.com

Jira Version	New API Version
7.2+	16.6.0

See Structure API Versions(see page 883) for full version information and downloads.

7.16.6.2 Compatible Changes in the Java API

Attribute API changes

Support for aggregation over subtree part has been added. Four subtree part kinds are supported:

1. `CHILDREN` – direct children only
2. `LEAVES` – leaf forest rows only
3. `STRICT` – full subtree without root node
4. `SUBTREE` – full subtree

Constants are declared in `CoreAttributeSpecs.Param`; behavior is described in `ReductionStrategy`. One can apply behavior to loader by subclassing `ReducingAggregateLoader` and overriding `ValueReducer.reduce` method (and other `ValueReducer` methods if convenient). Chosen subtree part is specified by `AttributeSpec` parameter 'type' (`CoreAttributeSpecs.Param.TYPE`) and defaults to `SUBTREE`.
List of subtree-part-aware loaders in api module:

- `LongSumLoader` (changed)
- `NumberSumLoader` (changed)
- `AbstractDistinctSumLoader` (changed)

To support this change some other changes have been made:

- Since distinct loaders can now accept parameters `DistinctAttributes.NON_UNIQUE_ITEMS_SPEC` has been deprecated and `DistinctAttributes` was marked as `@Internal`.
- Handy `SingleDependencyReducingAggregateLoader` (subclass of `ReducingAggregateLoader`) was added.
- Class `SecureSingleDependencyReducingAggregateLoader` aggregates values from subtree part, replacing value to `AttributeValue.undefined()` if user shouldn't have access to data by security reasons.
- `AttributeLoaderSecurity` encapsulates security checks. Can be used when `SecureSingleDependencyReducingAggregateLoader` isn't applicable.
- `AttributeValue.isEmpty()` is a shortcut for `!AttributeValue.isDefined() && AttributeValue.getLoaderData(Object.class) == null`
- `AttributeValue.ofNullable()` is a shortcut for `value == null ? AttributeValue.undefined() : AttributeValue.of(value)`
- `NumberAccumulator.toNumber()` is a shortcut for `NumberAccumulator.toValue().getValue()`
- `AbstractAggregateLoader.firstChildError()` method traverses errors through `AttributeLoader.Aggregate`. Used to transmit formula errors (other loaders currently don't supply error values).

`ForestSpec`-sensitive attributes

Attribute values can now depend on the forest spec for which the attribute is requested. Two new methods have been added to `AttributeContext` to support this:

- `getBaseForestSpec()` returns the `ForestSpec`, if there is one;

- `getBaseStructureId()` returns the structure ID part of the base forest spec.

Also there is a new version of `StructureAttributeSpec.getAttributeValues()`, marked `@Internal`.

The Notes column introduced in Structure 4.1 now uses the base forest spec instead of the `structureId` parameter, which has been removed.

Jira Service Desk Request Type support

A constant and a static method have been added to support the Request Type from Jira Service Desk as a separate item type:

- `CoreItemTypes.REQUEST_TYPE`
- `CoreIdentities.sdRequestType()`

Miscellaneous

- Added `GenerationContext.getForestSpec()`, which returns the forest spec being generated.
- `ValueFormat.ANY`, introduced in version 16.5.0, is now supported by the REST API.
- Added new versions of `ForestAccessCache.getInvisibleRows()` which accept `LongSizedIterable` as input. The old methods accepting `LongList` are now `@Deprecated`, kept only for binary compatibility.
- `ArrayForest.moveSubtreeAtIndex()` now returns -1 if no move is needed, i.e. the subtree is already at the given location.
- `@NotNull` and `@Nullable` annotations have been added in multiple places to clarify nullability contracts.

7.17 Structure 4.2 Release Notes

27th of August, 2017

Structure 4.2 adds major Formula language improvements, custom reporting period for Work Logged column, and a way to see large values that don't fit the column width. Other updates and improvements are also included.

Download the latest Structure and Extensions
(see page 978) Structure Demo Server[319]

7.17.1 Version Highlights

- Formula Column language improvements.
- Filtering work logs by a custom period in the Work Logged column.
- Large texts or other values that don't fit the column width can now be displayed fully.

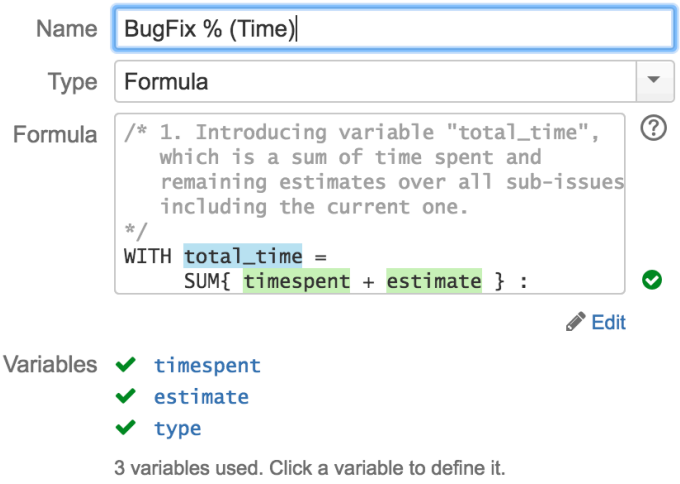
319 https://demo-structure.almworks.com/

7.17.2 Changes in Detail

7.17.2.1 Formula Language Improvements

There are a number of Expr language improvements delivered with this version.

First, we're adding **aggregate functions**, which are able to calculate a total value over sub-issues. In this version, we include only `SUM{}` aggregate function, but more aggregate functions will follow soon. But already with `SUM{}` only you can calculate a number of interesting metrics, such as WSJF or percentage of bugs in a structure. Here's a simple example of a formula that gives the number of bugs: `SUM{ IF(type=Bug;1) }`



Secondly, we're adding **local variables**, which is really helpful if the formula contains some expression multiple times.

Lastly, now you can write **comments** in a formula, which should make it is easier to read the formula later and make changes.

Remember that with Formula Column, it is possible to define a metric and share it with the team via Views(see page 336).

Documentation: Expr Language(see page 221), Aggregate Function Reference(see page 246)

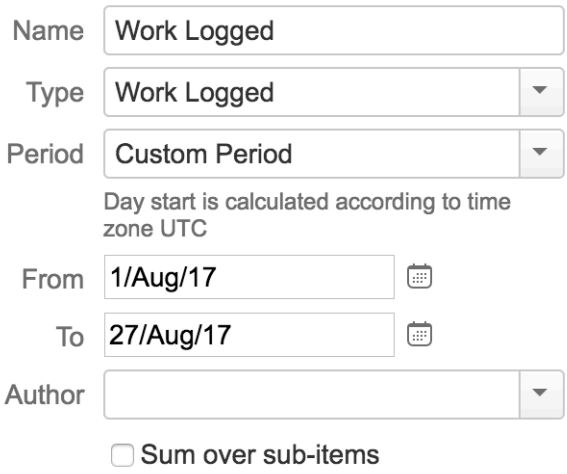
7.17.2.2 Custom Period in the Work Logged Column

Work Logged column displays total hours logged for an issue, with some additional filters. One of the filters selects only the work logs filed for a specific time period.

With Structure 4.2, it is possible to set an arbitrary period for Work Logged column (in addition to already available predefined periods like Today or This Year).

This addition option allows you to build task-based timesheet reports.

Documentation: Work Logged Column(see page 315)



7.17.2.3 Full Content Hover Box

When a field value is too large to fit into the cell, only a part of it is shown by Structure.

With version 4.2, it is possible to view the full content by hovering mouse pointer over the "more" sign (three vertical dots). This sign appears near the right edge of a cell if there's more to show.

This can come in handy if you do not want to click every issue to view their description in the issue details panel.

Documentation: Displaying Full Cell Content(see page 349)

7.17.2.4 Notable fixes and improvements

- The process of restoring Structure from backup now displays a progress bar and can be cancelled.
- Fixed: Changing filter text in the panel header does not result in change in filter results.
- Fixed: License installed on one node is not applied on another node in the cluster (JIRA Data Center).
- Fixed: Count leaves column has incorrect values when exported to Excel.
- Fixed: Values in Formula Cells with Duration format are missing when exported.
- Fixed: Potential data corruption during backup because of runtime exception.
- Fixed: Sub-task synchronizer doesn't add freshly created sub-tasks.
- Fixed: Time Tracking panel disappears when switching between issues on Internet Explorer.

The version also includes performance, reliability and other internal improvements.

7.17.3 Supported Versions

Structure 4.2 and all extensions support JIRA versions from 7.2 to 7.4.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

⚠️ Structure.Pages 1.2 or earlier are not fully compatible with Structure 4.2. If you're using Structure.Pages, make sure to also upgrade it to version 1.3 or later.

With respect to other add-ons and custom integrations, this release is backwards-compatible with Structure 3.4–4.1. Structure.Testy[320] extension, Colors[321] plugin, integrations with our partner add-ons should work with the new version.

7.17.4 Installation and Upgrade

7.17.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. ⚠️ If you have Structure.Pages installed, make sure you've upgraded to version 1.3 or later, both on JIRA and on Confluence side.
4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

320 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
321 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.17.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x-4.1 is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large JIRA, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.17.5 Enterprise Deployment Notes

Structure 4.2 has a number of changes that are particularly important for large installations and JIRA Data Center instances.

7.17.5.1 Events, Re-indexing and Experimental Feature

Over last few months, we've been working on tackling a race condition that may happen on a clustered JIRA. In short, Structure has its own means of inter-node communications, which is used to notify about issue changes. These notifications are important because changes made on one node may cause synchronizers or automation rules to be executed on another node. These rules or synchronizers often access JIRA Lucene index, to run a query, for example.

The problem is a race between our notification subsystem and JIRA's own notifications between nodes, which cause issue to be re-indexed on all nodes. If JIRA re-indexing is too late and our recalculation happens first, the users working with that node may see outdated or inconsistent data.

The solution to this problem required a lot of changes to Structure's event subsystem. Since it's a core component right in the middle of Structure's architecture, we approached its rollout carefully. Currently, the new event distribution approach for Data Center is an experimental "dark feature", which means that you need to turn it on explicitly, if needed.

To activate the updated "index monitoring" based event distribution subsystem on JIRA Data Center:

1. Verify that you need it. The cause for activating it could be the users complaining about not seeing the effects of actions of other users, or if your testing reveals this problem.
2. Set "`structure.delegatingItemTracker.enableReindexMonitor`" system property to "`true`".
3. Disable and then enable Structure. (Make sure there's enough time for disabling to propagate through all nodes.)
4. Verify: a new, undeletable system custom field called "**Structure Index Monitor**" should appear. This field will not have effect on issues, and it will disappear on its own if you turn this dark feature off.
5. Note that JIRA will tell you that a Full Re-index is required (since a new field added). Feel free to ignore this request, unless you also have other reasons for re-indexing.

In one of the upcoming versions of Structure, we'll enable this dark feature by default.

Takeaways:

- Current versions of Structure, including 4.2, in certain cases may show outdated data to some users on JIRA Data Center.
- The solution is currently an experimental feature, which can be enabled through a system property.
- There's an automatically created system custom field if this feature is enabled. It should not interfere with JIRA configuration.
- Even if the experimental feature is not enabled, there have been certain changes in a core subsystem in Structure.

7.17.5.2 Cancelling Structure Restore

Structure restore may take a long time, if there's a lot of Structure data (especially, history records). It may take an hour or more to restore Structure from backup, unless you decide to skip restoring history data.

With Structure 4.2, you can cancel the restore process if it takes too long. However, please be aware that after that you'll have some of the Structure data restored and some of it missing. You may need to communicate that fact to the users or adjust the access to Structure.

As an alternative, you can try Merge Structure operations, which works like Structure Restore but allows you to pick which structures and other data to restore.

7.17.5.3 Formulas and Attribute Subsystem

Attribute subsystem in Structure is responsible for delivering column values to the grid in Structure Widget, and for a few other things. In Structure 4.2 there were a few internal improvements in the Attributes subsystem, which may affect performance and functionality. (Actually, some of the changes should increase performance.)

One of the big features that we're working on over last several releases is Formula Column. This is an important addition to Structure that brings capability to define and run arbitrary metrics over structures of issues. Structure 4.2 introduces important additional functionality to formulas.

The expansion of formula feature may invite users to create new automated structures and running a formula with aggregation over those structures. On large JIRA instances this may lead to increased server load, as formulas with aggregation may need to scan a lot of issues to calculate a total.

Previously, we've added a hard limit on a structure's size (100,000 rows – can be adjusted). Calculating a total over 100,000 issues would require JIRA to retrieve that many issues from the database, which is not going to be quick and would add a load on JIRA and the database. It's a good idea to advise the users to calculate metrics on such large structures only if there's a real need. In most cases, structures can be made smaller. When in doubt, feel free to reach out to support@almworks.com[322].

7.17.5.4 Testing on Staging Environment

You can try load testing and stress testing Structure 4.2 and JIRA on a staging environment before upgrading.

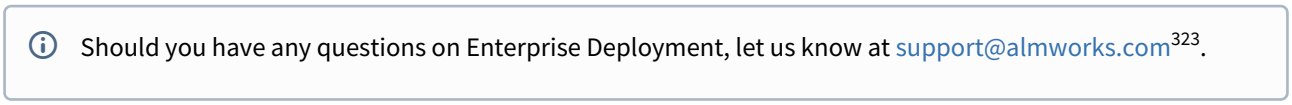
It makes sense to test the following parts:

- Open large structures and add multiple predefined columns from Calculated section. (See the description above about Formula Column feature.) Do that from a number of client computers using different user accounts.

322 mailto:support@almworks.com.

- Open the same structure in different browsers, having logged in under different users. Make sure the users go to different nodes in cluster. (Use direct node's addresses if necessary.) Make updates to a test structure and test issues in one browser and watch if changes are picked up (without refreshing the page) on another browser. (You might need to switch to the other browser to see the changes.)

The usual load and stress testing can also be applied.


Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[323].

7.17.6 API Changes in Structure 4.2

7.17.6.1 Minor Java API Release

There are a few moderate changes coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

JIRA Version	New API Version
7.1+	16.5.0

See Structure API Versions(see page 883) for full version information and downloads.

7.17.6.2 Compatible Changes in the Java API

Bulk attribute loading

`BulkAttributeLoader` is a new optional interface for attribute loaders. If your loader implements it, Structure will call its `preload()` method once for each attribute calculation, allowing you to perform efficient bulk calculations, e.g. run a single JQL query for all issues in the forest.

Three new methods have been added to `CompositeAttributeLoader` to support bulk attribute loaders: `hasBulkLoaders()`, `isBulkLoader()` and `preload()`.

Progress and cancellation for long-running processes

`ProgressGauge` is a new interface allowing you to track the progress of a long-running operation and gracefully cancel it from a different thread. Currently the only long-running operation with progress and cancellation support is `RestoreOperation`, which has a new method, `getProgressGauge()`. We plan to add progress and cancellation to `BackupOperation` in the future.

`ProcessHandleManager` implementation has been updated to support progress reporting and cancellation, both within a single JIRA instance and between different nodes in a Data Center environment. The following API methods have been added for this purpose:

- `ProcessInfo.getActivity()`
- `ProcessInfo.getPercentComplete()`
- `ProcessInfo.cancel()`

323 mailto:support@almworks.com

- `ProcessFeedback.isCancelled()`
- `ProcessUIController.setProgress()`

`SQuerySkeletonFactory` interface (experimental)

`SQuerySkeletonFactory` is an experimental extension point which lets developers add new query types to Structure. It is a work in progress, and it does not have client-side support yet.

Miscellaneous

- Added two new constants referring to the icon attribute, `CoreAttributeSpecs.ICON` and `CoreAttributeSpecs.Id.ICON`.
- Added a new attribute value format, `ValueFormat.ANY`.
- Added `getProjectsForCurrentUser()` method to `StructurePluginHelper`.
- Added `CoreSemantics.GROUP`, a new semantics for group rows created by groupers.
- Added `I18nText.setArguments(Object[])`, deprecated `setArguments(String[])`.
- Added `StructureUtil.isSubMap()` method.

`@Internal` components and classes

`ForestAccessCache` is the component used by Structure and Structure-based add-ons to check users' access to JIRA issues and Structure rows. It has been moved to the API and marked `@Internal`.

`SimpleAttributeProvider` is an abstract base class for several `AttributeLoaderProvider` implementations in Structure. Its implementation has been changed to better support optional spec parameters, and the class has been marked `@Internal`.

7.17.6.3 Item Type API Example

We have added a new sample plugin, `custom-itemtype`, which contains the following:

- a StructureItemType implementation for JIRA projects, complete with change tracking and access checks;
- an AttributeLoaderProvider implementation, which loads project names and icons;
- a StructureGenerator.Inserter implementation, which adds projects from one or more project categories.

You can download the plugin and its source code from API Usage Samples(see page 965).

7.18 Structure 4.1 Release Notes

19th of June, 2017

Structure 4.1 introduces Notes column for additional comments on any items in the structure and a Full Screen mode. It also contains other improvements, important updates and a security fix.

Download the latest Structure and Extensions
(see page 978) Structure Demo Server[324]

324 https://demo-structure.almworks.com/

7.18.1 Version Highlights

- Notes column lets you store additional text for any issue, folder or other item in a structure without creating custom fields or otherwise altering JIRA configuration.
- Grouping by version name and sprint name is useful for analysis when you have multiple projects with synchronized versions or sprints.
- Full-screen mode gives you more screen space for data and removes visual noise.

Structure 4.1 also contains a critical security patch.

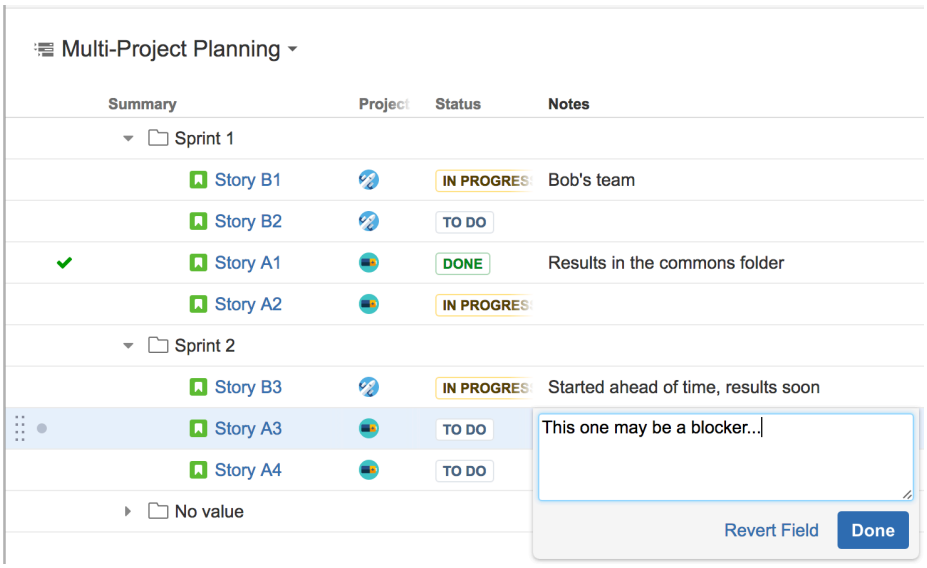
Upgrade is required for all JIRA instances running JIRA 7.1.0 and later. If you are using a previous version of JIRA, please make sure to upgrade to a version with the latest security patch – Structure 3.5.1(see page 714) for JIRA 7.0.x, Structure 3.3.5(see page 731) for JIRA 6.4.x, and Structure 3.2.3(see page 738) for JIRA 6.3.x.

> ⚠ You need to have a Structure license with active maintenance (expiring not earlier than June 20th, 2017) to run use the new version or the patches.

7.18.2 Changes in Detail

7.18.2.1 Notes Column

A new type of column - Notes - can now be added to a structure. It allows leaving additional notes in a structure for any issue or folder, without the need to create a custom field. Notes can be used to add status updates without disturbing the usual workflow for other users.



Each structure has its own notes, even for the same items.

Notes are stored in the Structure's database, so even if you remove Notes column, they are not gone – you can add the column back later and the notes you left are preserved.

Documentation: Notes Column(see page 319)

Keyboard shortcut: **Ctrl-Shift-|** (Windows) or **Command-|** (Mac) to add a column

7.18.2.2 Grouping by Version and Sprint names

It is now possible to group by version name (from Fix Version field) and sprint name.

This comes useful when you have multiple projects with a synchronized schedule, having the same versions and same sprints. If you combine issues from those projects together and group them by Sprint or by Fix Version, you'll get multiple groups for each sprint – because, from JIRA's standpoint, those are different things.

Grouping by the name of version or sprint is a case of grouping by text value (Text Attribute grouper is used). That grouper just considers the text value – in our case, the name of the sprint or version.

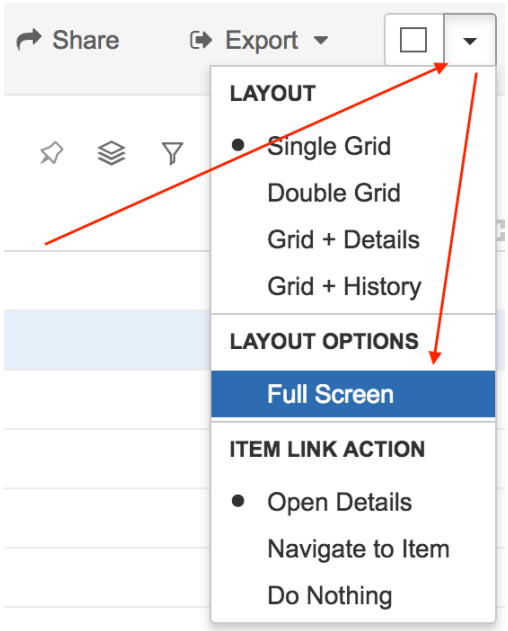
7.18.2.3 Full-Screen Mode

You can now expand Structure board to take the full screen, hiding JIRA header and, optionally, collapsing Structure toolbar. If you combine this with your browser's Full-Screen mode, you'll get maximum screen real estate dedicated to your tasks in a structure.

This is useful for presentations, using Structure board as a dashboard, working with large structures or just removing the visual clutter to concentrate on work.

Documentation: Full Screen Mode(see page 353)

Keyboard shortcut: **Z**



7.18.2.4 Structure Size Limit

We have introduced a hard limit on the structure size to safeguard against misconfigured synchronizers and imports. The limit is 100,000 rows maximum in a single structure, and it can be adjusted by a JIRA administrator.

A user who has Permission to Manage Synchronizers(see page 522) can inadvertently configure a synchronizer to pull in all issues into a single structure. That's not a problem unless JIRA instance is large. A structure with more than 100,000 issues in it may slow down JIRA, so we introduced this hard limit to protect the whole instance from erroneous actions of one user. Any action that would bring structure size over the limit would be denied.

Note that this limit only affects "static" structure, which does not include generators' output. Therefore, it protects from runaway synchronizers and manual changes. Generators have their own configurable limits and protections.

The hard limit may be adjusted by setting a system property and re-enabling Structure.

Documentation: Advanced Parameters(see page 690)

7.18.2.5 Security Patch

We have addressed a critical security issue, affecting all Structure versions starting with 3.0.

Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Security patches are provided separately for previous Structure versions, compatible with JIRA versions 6.3 and later.

7.18.2.6 Notable fixes and improvements

- Additional caching of sub-task links and issue links, improving performance of generators.
- "S" accelerator key removed from JIRA menu to avoid conflicts with Submit keyboard shortcuts in JIRA.
- Reduced number of log messages about deleted synchronizer when history is accessed.
- Fixed: Problems when deleting large structures from the user interface.
- Fixed: "After Filtering" option in aggregating columns is ignored by Export (both to Excel and Printable page).
- Fixed: Structure Gadget adds the displayed structure to Recent Structures list.
- Fixed: "Number of Leaves" column has no values exported in Excel.

7.18.3 Supported Versions

Structure 4.1 and all extensions support JIRA versions from 7.1 to 7.3.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

This release is backwards-compatible with Structure 3.4–4.0. Structure extensions – Structure.Pages[325], Structure.Testy[326], as well as Colors[327] plugin and other plugins that integrate with Structure 3.4 or later should work with the new version.

7.18.4 Installation and Upgrade

7.18.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.18.4.2 Upgrading Structure

> ⚠ If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x-4.0 is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)

325 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
326 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
327 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large JIRA, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.18.5 Enterprise Deployment Notes

Compared to Structure 4.0, Structure 4.1 has a couple important additions on the server side.

7.18.5.1 About Structure Size Limit

As described above, by default the maximum number of rows that can be placed into a structure manually, via synchronizer, via import or via API is 100,000. We introduced this limit following a couple of cases where JIRA performance degraded due to misconfigured synchronizers. This is an additional safeguard for large JIRA instances.

However, there might be legitimate cases where a structure needs more than 100,000 issues in it and JIRA environment is powerful enough to work with it. In that case, the limit can be lifted as described in Advanced Parameters(see page 690).

7.18.5.2 New Table for Per-Structure, Per-Item Properties

With the introduction of Notes column, described above, we have added AO_8BAD1B_S_ITEM_PROPERTY table to the database schema. This table stores primary data (not stored anywhere else in JIRA).

The data from this table is exported along other structure data when you do Structure Backup(see page 485) in a human and machine readable form. The table is also exported along with all Structure data during JIRA backup.

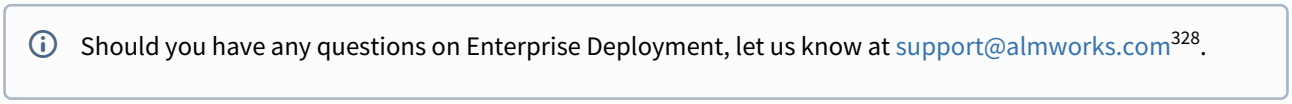
We don't expect considerable database load on this table.

7.18.5.3 Testing on Staging Environment

There are no particular special areas of interest for load testing and stress testing Structure 4.1. We advise running the same testing procedures as you've done for previous upgrades.

Given the changes described above, you can also test the following:

- Try creating a synchronizer that inserts all issues in JIRA into a test structure. Use Filter synchronizer and JQL "project is not empty". It should succeed up to 100,000 issues.
- Try using a moderately large structure (up to 10,000) rows, add Notes column, enter some values and sort by it. That would place some load on the database to download values for all items.



328 mailto:support@almworks.com

7.18.6 API Changes in Structure 4.1

7.18.6.1 Minor Java API Release

There are a few moderate changes coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

JIRA Version	New API Version
7.1+	16.4.0

See Structure API Versions(see page 883) for full version information and downloads.

7.18.6.2 Compatible Changes in the Java API

New methods in export API

Two new methods have been added to the export API to support the "after filtering" option in aggregated columns – `ExportRequestContext.requireAttribute(AttributeSpec, boolean)` and `ExportRow.get(AttributeSpec, boolean)`.

Another new method, `ExportRequestContext.getForestSpec()`, gives the export renderer provider access to the actual `ForestSpec` being exported.

`ItemTypeRegistry` interface

`ItemTypeRegistry` allows you to convert string item type IDs like `"com.almworks.jira.structure:type-issue"` to numeric IDs for more efficient storage and transport. Structure uses this registry internally, and now it's been made public.

Please note that the numeric IDs generated by `ItemTypeRegistry` are valid only on the JIRA instance they were generated on (including JIRA Data Center – they are the same for all nodes in a cluster). You cannot transfer them between different JIRA instances, you'll need to use the long string IDs for that.

Description property in `GeneratorPreset`

We've added optional descriptions to generator presets, so there's a new constructor `GeneratorPreset(label, description, parameters)` and a new method `GeneratorPreset.getDescription()`. Descriptions, if present, are shown as tooltips in the user interface.

Miscellaneous

- Added `equals()` and `hashCode()` implementations to `HistoryEntry` and `HistoryEntry.Change`.
- Added `info()` methods to `ConsiderateLogger`.
- Added `encodeURIComponent()` to `StructureUtil`.

7.18.7 Structure 4.1.1 Release Notes

20th of October 2017

Structure 4.1.1 is a security patch release.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[329]

7.18.7.1 Patch Release

This is a patch release based on Structure 4.1. We have addressed a minor security issue, affecting all Structure versions starting with 4.1 for those who use Notes column. Details of the issue are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Upgrade is required for all JIRA instances running Structure version 4.1 and using JIRA 7.1.x.

If you're using JIRA 7.2 or later, please upgrade to Structure 4.3[330] or later.

You should have Structure license with active maintenance (expiring not earlier than October 20th, 2017) to upgrade.

7.18.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.18.7.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x and 4.x is simple:

329 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
330 https://wiki.almworks.com/display/structure2gmaster/Structure+4.3+Release+Notes

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.18.8 Structure 4.1.2 Release Notes

5th of May 2020

Structure 4.1.2 is a security patch release.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[331]

7.18.8.1 Patch Release

This is a patch release based on Structure 4.1. We have addressed a security issue that affects all Structure versions starting with 3.5. Problem was fixed in the latest (6.0) Structure.

Upgrade is required for all Jira instances running Structure version 4.1 and using Jira 7.1.x.

If you're using Jira 7.2 or later, please upgrade to Structure 5.1.1 or later.

You should have Structure license with active maintenance (expiring not earlier than May 6th, 2020) to upgrade.

7.18.8.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your Jira version!
2. When the Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the Jira administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.18.8.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

331 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade procedure from versions 3.x and 4.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.19 Structure 4.0 Release Notes

26th of April, 2017

Structure 4.0 brings Excel-like functionality to JIRA with the new Formula column.

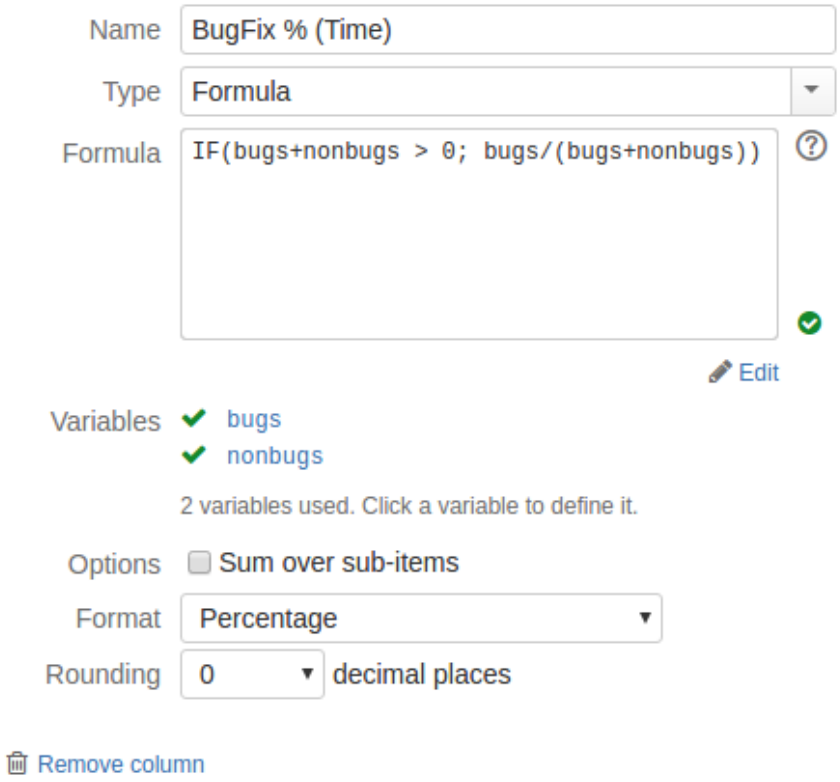
Download the latest Structure and Extensions
(see page 978)Structure Demo Server[332]

7.19.1 Version Highlights

- Formula column
- Work Logged column for a specific user
- Making content static when copying or cloning a structure
- Exporting structures to other add-ons

332 https://demo-structure.almworks.com/

7.19.2 Changes in Detail



7.19.2.1 Formula Column

With Formula column, any user can perform some calculation on every item in a structure and display the result in the Structure grid. It is very similar to how formulas are used in Excel or other spreadsheet software.

To add a formula to your Structure view, click "**+**" at the top right corner of the grid, just like when adding a new column of other type. Scroll down to the bottom of the list and select "**Formula...**" to write your own calculated expression, or select one of the metrics from "**Calculated**" section.

When configuring Formula column, you can define the calculated expression using Expr Language(see page 221). It allows you to use values coming from issue fields, Structure-provided attributes (like totals) and other formulas. The values can be combined together, using arithmetic operators and more than 70 functions, to arrive at a single numeric or text value. Not only is that value displayed in the Structure grid, but it can also be used to sort the structure (just click on the column header) or to group items with the same value together, using Text Attribute grouper(see page 698).

The potential applications for Formula column are truly limitless – from all kinds of metrics to status reports to cramming different information into a terse text. Structure 4 comes with just a few examples like "Percentage of Bugs" and "Weighted Shortest Job First" metric, but any user (not necessarily a JIRA administrator!) can set up their own formulas.

Documentation: Formula Column(see page 698), Expr Language(see page 221), Expr Function Reference(see page 226)

Keyboard shortcuts:

- **Ctrl+|** (or **Command+|** on Mac) to add a new column.

- **Shift+Enter** when editing a formula to have it saved and verified; hit **Enter** to continue editing.

7.19.2.2 Work Logged by User

The Work Logged column has been around for a while already – it shows total amount of time logged during a specific period (for example, during last week).

In Structure 4, this column has an additional "**Author**" option. By default, it is empty and the column shows the total work during the selected period for all users – as it did before. If a user is selected, then only work performed by that user is considered.

This lets you build a work distribution report for a small team – just add several Work Logged columns and select a specific user in each.

Oh, and, by the way, you can use the value from Work Logged column in a formula!

Documentation: Work Logged Column(see page 315)

7.19.2.3 Making Static Copy of a Structure

Manage Structure page in Structure 3 lets you create a copy of a structure. Additionally, you can opt to create clones of all issues in the structure and populate the new structure with clones. See Copying Structure and Cloning Issues(see page 368) for details.

Starting with Structure 4, you have an additional option that governs what happens with Generators(see page 698) and the dynamic content that they produce. (As a reminder, dynamic content is something that gets updated automatically when things change in JIRA. It is created by generators – see Automation(see page 122) feature description and Structure 3 Quick Start Guide(see page 698).)

You can decide to leave the same generators in the new structure (so they would provide similar content), remove generators altogether (which means removing dynamic content as well), or to remove generators but replace the dynamic content with a "static" copy. The latter option is especially interesting as it allows you to "freeze" a structure in time so that further changes in JIRA do not affect the composition of the structure.

Documentation: Copying a Structure(see page 365)

7.19.2.4 Exporting Structures to Other Add-ons

We have made it possible for other add-ons to provide additional options in the Export menu on the Structure Board.

At the moment, we have integrations with the following products:

- Pivot Report[333] add-on can be used for additional analysis of your issue data. The latest version already has integration with Structure, just open Structure Board and select **Export | Pivot Report**.
- Xporter[334] is a solution for exporting your issues to various formats, including Word documents and PDF. Starting with version 5.1, Xporter will add **Export | Xporter for JIRA** menu on the Structure Board that will let you create nice-looking documents with all issues from the structure. Headings in the document are arranged according to the hierarchy in the exported structure.

We are also working with a few other add-on vendors on more integrations.

---

333 https://marketplace.atlassian.com/plugins/biz.epicreport.epicreports/server/overview
334 https://marketplace.atlassian.com/plugins/com.xpandit.plugins.jiraxporter/server/versionhistory

7.19.2.5 Notable Fixes and Improvements

There are a number of issues addressed and improvements added in Structure 4.0. The following are worth mentioning:

- Fixed: "Show all sub-items of matching items" option for Filter Transformation doesn't work.
- Fixed: JQL Completion doesn't work in the query input.
- Fixed: "Exclude duplicates" setting is ignored during Export.
- When migrating from Structure 2.x, avoid long-running backups by turning off the "Backup History" option.

7.19.3 Supported Versions

Structure 4.0 and all extensions support JIRA versions from 7.1 to 7.3.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

This release is backwards-compatible with Structure 3.4–3.6. Structure extensions – Structure.Pages[335], Structure.Testy[336], as well as Colors[337] plugin and other plugins that integrate with Structure 3.4 or later should work with the new version.

7.19.4 Installation and Upgrade

7.19.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.19.4.2 Upgrading Structure

If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large JIRA, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

---

335 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
336 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
337 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.19.5 Enterprise Deployment Notes

Structure 4.0 does not have a lot of core changes compared to Structure 3.6.

7.19.5.1 Formula Column Notes

The largest feature – Formula Column – is completely new and will not affect how the rest of the product is working. However, the feature itself is open to all users – anybody can add a formula and start calculating the values for structures. The Expr language engine is very fast, however, a lot of calculations on large structures may result in general slowdown due to constant retrieval of issue values. This is an unlikely event though, and if there's a really large structure that could cause such trouble (100,000 issues or more), it would produce the load on JIRA server regardless of the Formula column.

7.19.5.2 Access to User Properties

Structure 4 has the ability to extract custom user properties, defined in the JIRA's user management section. In a vanilla JIRA installation, there are no properties, but one can use them for defining, say, the hourly rate of a user or a category of a customer. These properties can be used in Formula columns, based on some user field.

The side effect of this improvement is that user properties become visible to everybody, because anyone can use Formula or Attribute column to display them. If you'd like to keep the users from gaining access to these user properties and disable attributes that provide them for the formulas, you can turn on `structure.userPropertyAttribute.disable` system property. (Add `-Dstructure.userPropertyAttribute.disable=true` to startup script and use `System.setProperty("structure.userPropertyAttribute.disable", "true")` in Script Runner on a running JIRA, then re-enable Structure.)

7.19.5.3 Copying and Cloning Structures

The updated Copy / Clone feature provides more opportunities to start cloning a large number of issues. Like in other cases, this may be a problem if a structure is large enough or has an unbounded generator that produces a lot of issues.

The feature has sufficient warnings to the user, notifying them about the number of issues that are about to be cloned. If, however, you need to turn Copying/Cloning off for your JIRA instance, please let us know and we'll provide instructions about how to do that.

7.19.5.4 Backup

Please create Structure backup before upgrading. Turn off Backup History option unless you really need it.

7.19.5.5 Testing on Staging Environment

You can try load testing and stress testing Structure 4.0 and JIRA on a staging environment before upgrading.

It makes sense to test the following parts:

- Open large structures and add multiple predefined columns from Calculated section. (See the description above about Formula Column feature.) Do that from a number of client computers using different user accounts.
- Run a copy & clone of a large structure with dynamic content.

The usual load and stress testing can also be applied.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[338].

7.19.6 API Changes in Structure 4.0

7.19.6.1 Minor Java API Release

There are a few moderate changes coming with this release. The changes are backwards-compatible, so any code using API 16.x should work fine.

JIRA Version	New API Version
7.1+	16.3.0

See Structure API Versions(see page 883) for full version information and downloads.

7.19.6.2 Compatible Changes in the Java API

`BigDecimal` and `BigInteger` support for NUMBER attributes

Number attributes (see `ValueFormat.NUMBER`) are allowed to provide instances of `BigInteger` and `BigDecimal` as a result. The appropriate support was added to `NumberAccumulator` for calculating totals. We're limiting the precision of the numbers to `MathContext.DECIMAL64`.

Note that if your code used NUMBER attributes before and assumed that the result is either `Long` or `Double`, this assumption may be incorrect now.

Additional methods for formatting data in export API

The following methods were added to provide more power to the exporting code:

- `ExportColumn.setRounding()`
- `PrintableColumn.setDurationAsCalendarTime()`

Miscellaneous

- Additional factory methods in `CoreIdentities`.

7.19.6.3 Web Items

Extendable Export Menu

We have added a new web item location that lets you insert web items into Structure's Export menu.

338 mailto:support@almworks.com

Location name: `structure.export.menu`

To add a menu item, declare a `<web-item>` in that location and declare the following sub-elements:

- `label` - the name of the menu item, supports i18n, mandatory.
- `link` - the URL of the target page, mandatory. You can use `${forestSpec}` in the URL, which will be replaced with the URL-encoded JSON representation of the forest spec currently opened. If your URL does not have `${forestSpec}` in it, then forestSpec will be added to the URL as a query parameter.
- `tooltip` - the tooltip shown when mouse hovers over the item, supports i18n, optional.
- `styleClass` - a space-delimited CSS class names for the <i> icon element before the name, optional, defaults to a Structure-provided icon.
- `weight` attribute - defines the order of appearance of the links, optional.
- `condition` / `conditions` - can be used to display the links conditionally.
- `context-provider` - can be used to add additional variables to the link labels or URL.

7.20 Structure 3.6 Release Notes

4th of April, 2017

Structure 3.6 is another step towards integration with third-party add-ons. Now Structure supports grouping by Account field from Tempo Timesheets[339]. Apart from that, the release contains a number of fixes and smaller improvements.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 742) before upgrading.

Download the latest Structure and Extensions
(see page 978) Structure 3 Demo Server[340]
Structure 3 Quick Start Guide
(see page 704)

7.20.1 Version Highlights

- Grouping by Account field from Tempo Timesheets
- Support for Portfolio 2.2.3 and 2.2.4

7.20.2 Changes in Detail

7.20.2.1 Grouping by Account

When building your structures you can now easily track progress and see distribution of time spent, remaining estimates and other values by Accounts that you have set up in Tempo Timesheets. To do that, simply add a Group Generator for Tempo Account, select if you want to see closed and archived accounts and if you want to be able to

339 https://marketplace.atlassian.com/plugins/is.origo.jira.tempo-plugin/server/overview
340 https://demo-structure.almworks.com/

move issues between accounts in structure. You can use this grouping both in Automation (when building a structure) and in Transformations.

Structure supports Tempo 8.2.4/Tempo Account 3.0.7.

7.20.2.2 Support for Portfolio 2.2.3 and 2.2.4

The previous version introduced support for Portfolio Parent links[341] (both Group and Extend). This version adds compatibility with the latest versions of Portfolio.

7.20.2.3 Notable Fixes and Improvements

The following issues have been addressed in Structure 3.6:

- Fixed: Structure Gadget in Confluence becomes blank and cannot be configured.
- Fixed: Groups are grayed out, when applying Filter transformations.
- Fixed: Changes made to generators parameters are sometimes not applied.
- When cloning a structure with Automation, statistics is shown for the number of static items and items added dynamically.

The new version also contains other bug fixes and improvements.

7.20.3 Supported Versions

Structure 3.6 and all extensions support JIRA versions from 7.1 to 7.3.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

This release is backwards-compatible with Structure 3.5, so all Structure extensions – Structure.Pages[342], Structure.Testy[343], as well as Colors[344] plugin and other plugins that integrate with Structure 3.5 – should work with the new version.

7.20.4 Installation and Upgrade

7.20.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

341 http://almworks.com/blog/2017-01-track-portfolio-plans-real-time.html
342 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
343 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
344 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

7.20.4.2 Upgrading Structure

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Back up Structure data. Use **Administration | Structure | Backup Structure** menu item. If you have a lot of structures and a large JIRA, consider turning off "**Backup History**" option to avoid long backup process.
3. Install the new version of the plugin.
4. Upgrade Structure.Testy[345] and Structure.Pages[346] add-ons if you're using them.
5. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.20.5 Enterprise Deployment Notes

Structure 3.6 has a few changes in the core parts that could potentially affect performance and stability of the system. The changes are part of the ongoing improvement to optimize memory and database footprint by having a procedure of removing unused "rows" from Structure. This optimization itself is not yet turned on in Structure 3.6.

Rigorous testing has been carried out on the release candidate. The suggestions below provide further means to assure seamless deployment.

7.20.5.1 Backup

Please create Structure backup before upgrading. Turn off Backup History option unless you really need it.

7.20.5.2 Testing on Staging Environment

You can try load testing and stress testing Structure 3.6 and JIRA on a staging environment before upgrading.

It makes sense to test the following parts:

- Loading JIRA's issue page for many issues by many users at once. (Use issues from projects that are enabled for Structure.)
- Loading different structures and queries on the Structure Board by many users at once.
- Creating structures, populating them with generators and then deleting them.
- Creating a structure, adding considerable amount of issues (for example, 10,000 issues) and removing them from the structure.

The usual load and stress testing can also be applied.

345 https://marketplace.atlassian.com/plugins/com.almworks.testy
346 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[347].

7.20.6 API Changes in Structure 3.6

7.20.6.1 Minor Java API Release

With this release we add a few constants and methods to the Java API. The changes are backwards-compatible.

JIRA Version	New API Version
7.x	16.2.0

See Structure API Versions(see page 883) for full version information and downloads.

7.20.6.2 Compatible Changes in the Java API

Tempo Account support

We have added two constants and two static methods to support the new Tempo Account grouper and the corresponding built-in item type:

- `CoreStructureGenerators.GROUPER_TEMPO_ACCOUNT`
- `CoreItemTypes.TEMPO_ACCOUNT`
- `CoreIdentities.tempoAccount(int accountId)`
- `CoreIdentities.isTempoAccount(ItemIdentity itemId)`

New versions of `RowManager.findRows()`

We have added two new `findRows()` methods to the `RowManager` interface:

- `void findRows(ItemIdentity itemId, LongPredicate consumer)`
- `default void findRows(ItemIdentity itemId, LongConsumer consumer)`

These methods guarantee that all row IDs your `consumer` is given during the `findRows()` call can be resolved by `getRow(long)` without throwing a `MissingRowException`. You can use the first method to stop the scan early.

The original method `LongIterator findRows(ItemIdentity itemId)` has been converted to a default method. Please note that this method will scan all rows before returning and that the row IDs produced by the resulting `LongIterator` may get deleted after the `findRows()` call returns.

347 mailto:support@almworks.com

7.21 Structure 3.5 Release Notes

30th of January, 2017

Structure 3.5 is a major release that adds grouping by issue links, filtering by JQL or S-JQL in Structure Gadget and bidirectional links extender. This release also has quite a few other updates, important security fixes, performance and stability improvements.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 742) before upgrading.

Download the latest Structure and Extensions
(see page 978) Structure 3 Demo Server[348]
Structure 3 Quick Start Guide
(see page 708)

7.21.1 Version Highlights

Structure 3.5 is a major update in the Structure 3 series. It contains several new features:

- Grouping by issue links
- Specifying JQL or S-JQL as a filter in Structure dashboard gadget
- Bidirectional issue link extender
- Default level limit for extenders

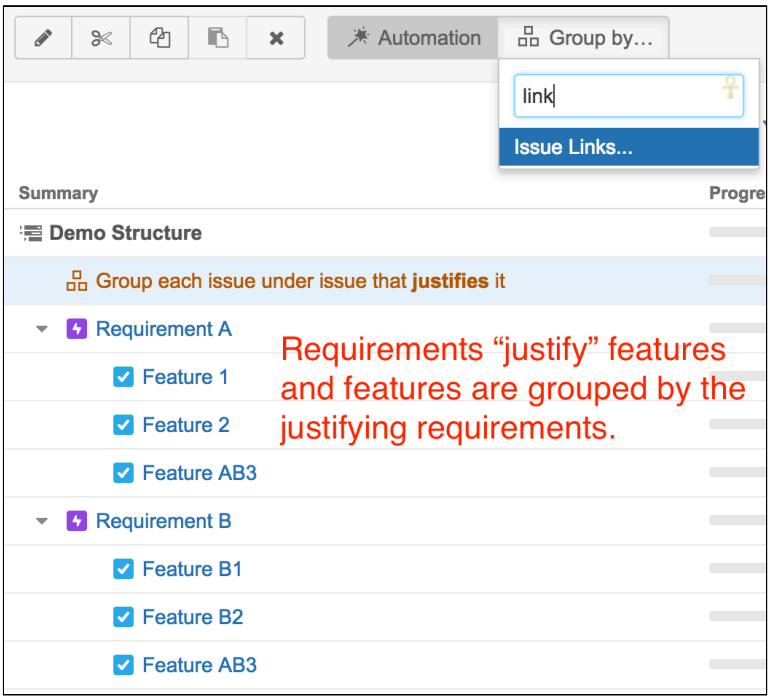
This version also contains a critical security patch.

Upgrade is required for all JIRA instances running Structure 3.0 and using JIRA 7.x. If you are using a previous version of JIRA, please make sure to install a version with a security patch – either Structure 3.3.4(see page 730) for JIRA 6.4.x or Structure 3.2.2(see page 737) for JIRA 6.3.x.

You need to have a Structure license with active maintenance (expiring not earlier than January 30th, 2017) to run use the new version or the patches.

348 https://demo-structure.almworks.com/

7.21.2 Changes in Detail



7.21.2.1 Grouping by Issue Links

There's a new grouper that can arrange a list of issues based on the incoming links of a certain type.

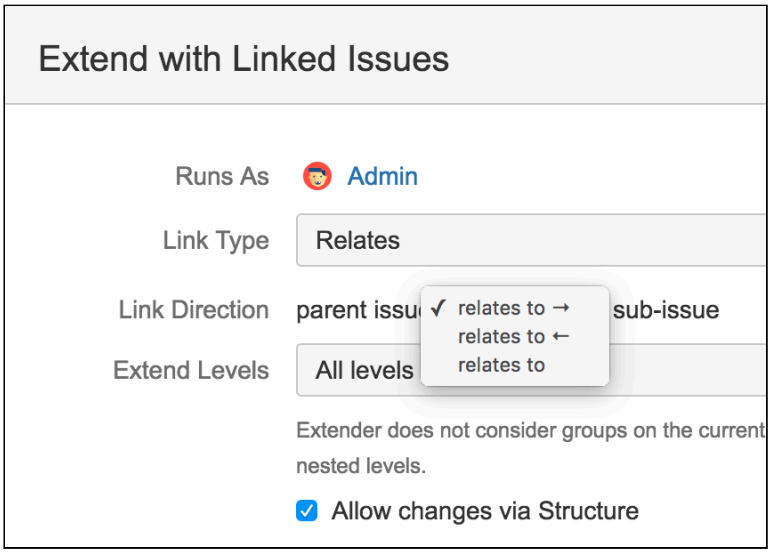
For example, if you have a list of Features and there are links between Requirements and Features, you can have the Features placed under the Requirement that is linked to it. If there are two incoming links from different Requirements, the Feature will be placed under both parent items.

7.21.2.2 Improved Filtering in Structure Gadget

Before Structure 3.5, the content shown in a Structure dashboard gadget could be filtered with a JIRA's saved filter. In this version we made it possible to also filter by a text query, a JQL query or an S-JQL query.

Support for S-JQL is especially important here, because it allows you to see a part of a big structure based on a hierarchical query. For example, you can display only "critical" issues and all their children, or a contents of a specific folder.

Documentation: Structure Gadget(see page 394), S-JQL Cookbook(see page 267)



7.21.2.3 Bidirectional Issue Link Extender

Sometimes a link type does not have a "direction". An example is the standard JIRA's link type, "relates to". However, technically, there are two directions, but they are named identically. This was making it hard to build a hierarchy of related issues, because you would have to add a separate extender for each direction.

Starting with Structure 3.5, you can select both directions in the Links Extender, and it will produce a clean hierarchical view of the linked issues, regardless of the directions of the links.

The newly introduced Issue Links Grouper also support bidirectional operation.

7.21.2.4 Default Level Limit for Extenders

All extenders have "Levels" setting, which govern on what levels down the hierarchy the extender is applied.

Without a limit, the extender is applied at all levels under the parent that contains the extender. This sometimes led to problems where the extender was not configured properly and it resulted in structures 100 levels deep or more.

In Structure 3.5, the default setting for all extenders is to apply to 10 levels under. Should you need to increase that number, just edit the extender settings.

7.21.2.5 Security Patch

We have addressed two medium-to-critical security issues, affecting all Structure versions starting with 3.0.

Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

7.21.2.6 Notable Fixes and Improvements

The following issues have been addressed in Structure 3.5:

- JQL queries containing structure() function should execute faster.

- Backlog template now offers more fields to group by.
- Fixed: Structure does not react to issue updates after upgrading Universal Plugin Manager plugin.
- Fixed: Structure requiring unlimited commercial license in case one of the application licenses for JIRA is an evaluation license.

The new version also contains other bug fixes and improvements.

7.21.3 Supported Versions

Structure 3.5 and all extensions support JIRA versions from 7.0 to 7.3.x. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

> ⚠ Structure 3.5 is the last version of Structure to support JIRA 7.0. Structure 3.6 will most likely support JIRA versions starting from 7.1.

This release is backwards-compatible with Structure 3.4, so all Structure extensions – Structure.Pages[349], Structure.Testy[350], as well as Colors[351] plugin and other plugins that integrate with Structure 3.4 – should work with the new version.

7.21.4 Installation and Upgrade

7.21.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.21.4.2 Upgrading Structure

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
> - If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)

---

349 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
350 https://marketplace.atlassian.com/plugins/com.almworks.testy/server/overview
351 https://marketplace.atlassian.com/plugins/com.almworks.jira.colors.colors-plugin/server/overview

2. Install the new version of the plugin.
3. Upgrade Structure.Testy[352] and Structure.Pages[353] add-ons if you're using them.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.21.5 Enterprise Deployment Notes

Structure 3.5 is a fairly significant update. Some of the improvements are directly related to performance, stability and important Structure functionality.

7.21.5.1 Security Patches

Please note that the plugin contains security patches. It is important to upgrade to avoid potential data exposure. For the sake of safety, for time being we share the details only in the closed Structure Security Google Group[354].

If you cannot upgrade to Structure 3.5 because you have JIRA 6, please check out Structure 3.3.4(see page 730) and Structure 3.2.2(see page 737).

7.21.5.2 Plugin Start-Up Sequence

We have made further improvements to the plugin start-up sequence, where it needs to orchestrate the components start-up with JIRA start-up. Over time, this has proven to be tricky, but our current solution should be very robust. The improvements in Structure 3.5 concern the case where the UPM (Universal Plugin Manager) plugin is updated. This causes the whole plugin system to restart, and it is yet another case with different JIRA behavior.

Although we are pretty sure in the current solution, keep an eye out at the start of JIRA or when UPM is upgraded, watch the logs, and if there are any warnings or errors, please let us know.

7.21.5.3 Automation Engine Locking

We have made improvements to the Automation Engine, which should reduce the number of cases where you see timeouts in the logs ("cannot acquire lock in 30000ms") for various structures.

Please do not confuse this with other locks! JIRA has a lot of locks and many have 30 second timeouts. These locks in particular don't block other JIRA actions, but come into play when multiple users open the same structure (or when a single user opens the same structure in multiple browser windows). The locks protect JIRA server from doing the same work in parallel. If there is a timeout, it means that some structure is generated for a long time – for more than 30 seconds – so either Automation engine has to crunch a lot of issues, or the system is slow generally, or there is some configuration problem. In any case, you can inspect the structure, finding it by its ID, which is given in the same warning in the logs.

However, this change is in a critical and very complex part of the product, so there's a small risk of locking issues, related to the aforementioned warnings in the logs, structure lock-downs (when a structure is not displayed, the users just see empty content), or redundant calculations on the server resulting in higher server load.

352 https://marketplace.atlassian.com/plugins/com.almworks.testy
353 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
354 http://groups.google.com/group/structure-security-list

7.21.5.4 Asynchronous Index Writing

Structure keeps an index of "which structures contain which issues". This index is not critical to operation and is used only sometimes – for example, when selecting which structure to display on the issue page. The index also ignores the generated content; it only works for issues added manually or by synchronizers.

In some configurations, updating that index introduced unwanted latency, so we moved its updates to an asynchronous job, executed by a background thread. While this change is not likely to cause any trouble, the index updates may be delayed on a heavily loaded system and that may result in an incorrect structure being picked to show on an issue page.

7.21.5.5 Testing on Staging Environment

It is recommended to try Structure 3.5 on a staging environment before upgrading. The suggested tests are:

- Plugin lifecycle.
    - Installing and uninstalling. Disabling and enabling.
    - Upgrading UPM, upgrading JIRA Agile plugin.
    - Restarting JIRA. (There are no Data Center-specific changes in this version.)
- Try making changes to structures in rapid bursts. Watch out for JVM memory statistics and CPU consumption. See if there is a lot of activity even after the bursts stop.
- Check out generators that work with issue links – the Extender and the Grouper. Try using unidirectional and bidirectional configurations on large structures.
- Check Structure Gadget.
    - Check if it works on JIRA dashboard.
    - If you have Confluence instance connected and Structure gadget is used there, check that it works in your configuration.
- Check "Export" button, exporting to Excel and to Printable page. Try on large structure and under a user account without administrative privileges.
- Automation locking.
    - Try opening a structure with automation from multiple browsers. You can build a temporary structure with considerable work for generators – for example, inserting 5,000 issues and using Issue Links extender.
    - Try using transformations (such as filters) on a structure, along with opening that structure in other browsers.
- Automation load testing.
    - Try opening user structures that contain considerable amount of issues, 10,000 or more. Click grid header to make Structure sort structure by some field. Do that in multiple browser tabs and using different columns.
    - Try creating a new structure and populate it with the help of an Inserter (use Automation | + | Insert | JQL) and raise the issue limit to 10,000. Repeat this several times with different structures.
    - Watch log files for errors and warnings.
- Automation stress testing.
    - Emulate peak number of users opening the most popular structure.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[355].

355 mailto:support@almworks.com

7.21.6 API Changes in Structure 3.5

7.21.6.1 Minor Java API Release

With this release we add a few constants and a static method to the Java API. The changes are backwards-compatible.

JIRA Version	New API Version
7.x	16.1.0

See Structure API Versions(see page 883) for full version information and downloads.

7.21.6.2 Compatible Changes in the Java API

New constant in `StructureLicenseError`

- `JIRA_EVAL_LICENSE_MISMATCH`

New constants in `CoreStructureGenerators`

- `GROUPER_AGILE_SPRINT`
- `GROUPER_LINKS`

New method in `StructureUtil`

- `getBooleanSystemProperty(String key, boolean defaultValue)`

7.21.7 Structure 3.5.1 Release Notes

19th of June 2017

Structure 3.5.1 is a security patch release for JIRA 7.0.x.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[356]

7.21.7.1 Patch Release

This is a patch release based on Structure 3.5.0. We have addressed a critical security issue, affecting all Structure versions starting with 3.0. Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

356 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is required for all JIRA instances running Structure versions 3.0–3.5.0 and using JIRA 7.0.x.

If you're using JIRA 7.1 or later, please upgrade to Structure 4.1(see page 690) or later.

⚠ You should have Structure license with active maintenance (expiring not earlier than June 20th, 2017) to run the patches.

7.21.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.21.7.3 Upgrade

⚠
- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.21.8 Structure 3.5.2 Release Notes

ⓘ **5th of May 2020**

Structure 3.5.2 is a security patch release for Structure 3.5

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[357]

357 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.21.8.1 Patch Release

This is a patch release based on Structure 3.5.1. We have addressed a security issue that affects all Structure versions starting with 3.5. Problem was fixed in the latest (6.0) Structure.

Upgrade is required for all Jira instances running Structure versions 3.5.0 and using Jira 7.0.x.

If you're using Jira 7.1 or later, please upgrade to Structure 4.1(see page 690) or later.

> ⚠ You should have Structure license with active maintenance (expiring not earlier than May 6th, 2020) to run the patches.

7.21.8.2 Installation

If your Jira server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.21.8.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
> - If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up Jira data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with Jira data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.22 Structure 3.4 Release Notes

> ⓘ **12th of December, 2016**
>
> Structure 3.4 is a major release that adds quick transformations (a.k.a. quick filters), integration with Portfolio for JIRA and stable public API. It also has quite a few other updates, bug fixes, performance and stability improvements.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 742) before upgrading.

Download the latest Structure and Extensions
(see page 978) Structure 3 Demo Server[358]
Structure 3 Quick Start Guide
(see page 716)

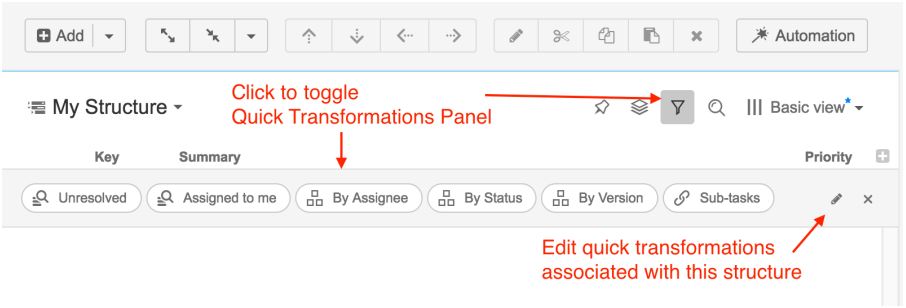
7.22.1 Version Highlights

Structure 3.4 is a major update in the Structure 3 series. It contains several new features:

- Quick Transformations
- Integration with Portfolio for JIRA
- Issue Details layout on the Project page
- Grouping by text fields
- Performance improvements
- Public API

Upgrade is recommended to all users.

7.22.2 Changes in Detail



7.22.2.1 Quick Transformations

Quick transformations, also known as "quick filters", are predefined transformations that may be configured by the structure owner and then quickly turned on, separately or in combination.

Previously, Structure had only two predefined quick transformations – Unresolved and Assigned to me. Now it is possible to define your own quick transformations and share them with the team.

Documentation: Quick Transformations(see page 188)

7.22.2.2 Integration with Portfolio for JIRA

We have added support for Parent Link field in Portfolio for JIRA.

358 https://demo-structure.almworks.com/

Now, when you have Portfolio installed, you get two new generators:

- An extender called **Child Issues (Portfolio)**, which adds "child" issues, as defined by Parent Link field, under their "parent" issues.
- A grouper called **Parent Issue (Portfolio)**, which goes the other direction – groups issues based on the value in their Parent Link field, which adds the parent issues to structure.

Both generators support updates, so you can change the Parent Link field by dragging and dropping issues.

This feature allows you to recreate in Structure the hierarchy you have in Portfolio and then build on top of it.

Supported Portfolio for JIRA versions: 2.1.5 or later.

7.22.2.3 Issue Details Layout on Project Page

Good news for those of you who prefer to use Project tab with Structure, rather than Structure Board – we have added "Grid + Details" layout to those available on the project page.

See Viewing Issue Details(see page 108) for documentation about this layout.

7.22.2.4 Grouping by Text Field

We have added a special generator that lets you group by a text attribute. A common example is grouping by the value of a custom text field.

To use this grouper, select **Automation | +**, then **Group**, and find **Text Attribute...** in the drop-down list. In the dialog that follows, select the desired attribute.

This grouper does not handle changes. You cannot move an issue from one group to another – you will need to update the field.

7.22.2.5 Performance Improvements

There are a few performance improvements in this version:

- We have replaced one of the core components, *RowManager*, that has been causing some trouble in the past. The new version should be much faster.
- There were significant fixes related to synchronizer performance.
- Backup creation with history is now much faster.
- We fixed some performance issues with structure() function used on Agile boards.
- Fixed performance issues caused by too many groups in the user directory.

7.22.2.6 Public API

We are finally releasing our stable API for Structure 3 series.

With the introduction of the new architecture earlier this year we had to make breaking changes to the API and took some time to work out the kinks and make it stable. Structure 3.4 ships with Structure API 16.0.0, which can be used in your custom plugins or from integration scripts.

We have documented the most important parts of the API but the documentation work still continues. You can expect that API coverage will increase with every new version.

Documentation: Structure Developer's Guide(see page 842)

7.22.2.7 Notable Fixes and Improvements

The following other issues have been addressed in Structure 3.4:

- Structure restore writes progress percent to the logs, so JIRA administrator can check it.
- Fixed: Compatibility issues with Zephyr plugin.
- Fixed: Incorrect sorting direction when adding a sorter, in certain cases.

There are also other bug fixes and improvements.

7.22.3 Supported Versions

Structure 3.4 and all extensions support JIRA versions from 7.0 to 7.2. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

> ⚠ Structure 3.3 was the last version of Structure to support JIRA 6. Structure 3.4 supports JIRA versions starting from 7.0.

7.22.3.1 Structure.Pages Upgrade Required

Please note that if you're using Structure.Pages, you will need to upgrade it to version 1.2.0.

See Structure.Pages 1.2 Release Notes[359] for more information about updates in Structure.Pages.

7.22.3.2 Structure.Testy Upgrade Required

Please note that if you're using Structure.Testy, you will need to upgrade it to version 2.2.0.

Structure.Testy 2.2 includes new high-level API that makes it easier to integrate your scripts with Testy and update test statuses programmatically.

7.22.4 Installation and Upgrade

7.22.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

---

359 https://wiki.almworks.com/pages/viewpage.action?pageId=32213737

7.22.4.2 Upgrading Structure

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Upgrade Structure.Testy[360] and Structure.Pages[361] add-ons if you're using them.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.22.5 Enterprise Deployment Notes

We now include an additional section in Release Notes that aims to address the concerns of deploying the upgrade at a large enterprise. In this section we will suggest how the changes in the new version may affect stability and performance of JIRA and provide ideas for testing the new version on a staging environment.

Structure 3.4 is a fairly serious update. A lot of code underwent refactoring, performance optimization and testing.

The following changes are quite important for the large-scale instances.

7.22.5.1 New implementation of RowManager component

We have rewritten the component that is central to the Structure's operation. The component's role is to store temporary rows as they are generated by Automation engine. The previous version was using "MapDB" library and offloading the stored values to disk. This turned out to be unreliable in some circumstances and under a heavy load.

The new version is much simpler and efficient, however, it does store this information in memory. The amount of data stored is not overly large, but current implementation never removes data from memory once it is stored. Therefore, on an active enough and large enough instance, this component may eventually grow to take a lot of memory resources or even cause an OutOfMemoryError.

Therefore, JIRA administrators on large instances are advised to keep an eye on the amount of Heap memory consumed by JIRA's Java process, and if it reaches what you'd consider to be a high threshold, you can clean up the memory taken by Structure with a simple operation: disable Structure plugin and then enable it back.

We are continuing our work on improving this component. Our next versions will contain improvements for this aspect.

360 https://marketplace.atlassian.com/plugins/com.almworks.testy
361 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview

7.22.5.2 Indexing on Data Center

We have identified a problem that can possibly manifest on JIRA Data Center cluster, when a node wants to recover indexes (from a downloaded index zip from another node), but it happens to be running a Structure synchronizer. As synchronizers used to take read lock on reindexing, they prevented full-stop reindex and index recovery from happening at the same time with synchronization. But on large instances a single synchronization job may execute for a long time – and this caused "Wait attempt timed out" error on the reindexing node.

The new locking mechanism, which is added in Structure 3.4, does not take read lock for more than a fragment of a second. While still ensuring that Structure's synchronizers and generators would execute based only on consistent query results, it allows JIRA full reindex or index recovery to happen at any time.

7.22.5.3 Improved Start-up Sequence

We have changed the way Structure plugin starts. Since we only support JIRA 7.0 or later with this version, we were to use some of the improvements Atlassian team prepared in JIRA 7 to increase reliability of the start sequence.

The related problem that happened to our customers in the past was that the plugin wouldn't start – it would wait for Active Objects component (provided by JIRA) to become available, but fail after 60 seconds of waiting. Usually, another attempt to manually start Structure would succeed.

With the new start-up sequence, the probability of such behavior should be less. Unfortunately, it's hard to tell precisely, because there's a dependency on JIRA subsystem. However, the plugin start-up has become much more predictable and repeatable.

7.22.5.4 Performance Improvements

Most of the performance improvements mentioned in the Release Notes above are quite important for large-scale instances.

7.22.5.5 Testing on Staging Environment

The following checks are suggested for Structure 3.4 on a staging environment with production data.

1. Start-up and shutdown.
    a. Try installing and uninstalling Structure several times. Watch the logs. On JDC, watch logs on every node.
    b. Try starting and stopping JIRA when Structure is installed. On JDC, try stopping/starting individual nodes and the whole cluster.
2. Index recovery on JIRA Data Center. Try forcing index recovery by manually delaying your node indexes. Before that, install synchronizers that are known to run for a long time – for example, a links synchronizer on all of your issues, assuming you have a lot of links in JIRA.
3. Automation load testing.
    a. Try opening user structures that contain considerable amount of issues, 10,000 or more. Click grid header to make Structure sort structure by some field. Do that in multiple browser tabs and using different columns.
    b. Try creating a new structure and populate it with the help of an Inserter (use Automation | + | Insert | JQL) and raise the issue limit to 10,000. Repeat this several times with different structures.
    c. Watch log files for errors and warnings.
4. Automation stress testing.
    a. Emulate peak number of users opening the most popular structure.

5. If you're using JIRA Software, try creating an Agile board that is based on `structure()` function (see S-JQL Cookbook(see page 267)). Make sure there is about 1,000 issues in the result. See how well Agile board loads.
6. Check **Administration | Structure | Maintenance** page. Try to run Structure backup and measure how much time does it take. Watch for errors in the logs.

Should you have any questions on Enterprise Deployment, let us know at support@almworks.com[362].

7.22.6 Structure 3.4.1 Release Notes

30th of December 2016

Structure 3.4.1 is a patch release based on Structure 3.4.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[363]

7.22.6.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has been changed in Structure 3!

This is a patch release based on Structure 3.4.

We have addressed a problem with Structure widget being shown on the issue page for issues from the projects for which Structure is not enabled.

Upgrade is recommended for all instances where Structure availability is set on the project basis.

7.22.6.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

362 mailto:support@almworks.com
363 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.22.6.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.22.6.4 Enterprise Deployment Notes

We now include an additional section in Release Notes that aims to address the concerns of deploying the upgrade at a large enterprise. In this section we will suggest how the changes in the new version may affect stability and performance of JIRA and provide ideas for testing the new version on a staging environment.

In terms of stability and performance, this patch does not bring significant changes compared to version 3.4.0.

7.23 Structure 3.3 Release Notes

9th of September, 2016

Structure 3.3 is a major release that adds support for structure migration between JIRA instances, extends S-JQL with ability to search for folders and other non-issue items, addresses compatibility issues with JIRA 7.2 and has other improvements and important fixes.

Important for Structure 2.x users! If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 742) before upgrading.

Download the latest Structure and Extensions
(see page 978) Structure 3 Demo Server[364]
Structure 3 Quick Start Guide
(see page 723)

364 https://demo-structure.almworks.com/

7.23.1 Version Highlights

Structure 3.3 is a major update in the Structure 3 series. It contains several new features:

- Migration of Structure data between JIRA instances (also allowing partial restore from backup)
- S-JQL support for searching folders and other items
- Special filter that removes duplicates added by a combination of inserters and extenders
- Maintenance task that removes old structure change history (please check the defaults!)
- Swapping panels in two-panel mode
- JIRA 7.2 compatibility
- Significant fixes and improvements related to performance and troubleshooting

Upgrade is recommended to all users.

7.23.2 Changes in Detail

7.23.2.1 Structure Data Migration

It's now possible to import one or more structures from a different JIRA instance after you have imported projects with the JIRA's Project Import operation. Also, you can partially restore deleted or broken structures from a backup without affecting other structures.

Documentation: Structure Backup, Restore and Migration(see page 485)

> Migration used to be a Structure 2 feature, but it was disabled with the release of Structure 3, as we've moved the storage from a separate database to the JIRA main database. Now it is being reinstated.

7.23.2.2 Search for folders and other non-issue items with S-JQL

It is now possible to identify folders (whether specific folders by name or all folders) and other types of items in S-JQL expressions. This expands the power of S-JQL to all items – for example, it allows searching for a contents of a specific folder.

For example: `descendant of folder("next release")`

Documentation: Structured JQL(see page 266)

7.23.2.3 Special filter for removing duplicates from using Inserter and Extender

It is a typical setup when you use JQL Inserter to place all matching issues at some place in a structure, and then use an extender (for example, Links Extender) to build a hierarchy of relationships starting from the inserted issues. For example, to show all issues from a project with dependencies placed hierarchically, you'd add a JQL Inserter (`project = ABC`) and a Links Extender with (`depends on`) link type.

The problem with such setup is that an issue may get inserted once by the inserter, and then again added at a deeper hierarchy level by the extender.

To remove such duplicates, use a special "Remove Inserter/Extender Duplicates" filter.

Documentation: Inserter/Extender Duplicates Filter(see page 133)

7.23.2.4 Maintenance task to clear old change history

Daily maintenance now includes clearing up database space by removing old change history for structures. It is recommended to not disable this task to avoid database size buildup and reduction in activity streams querying performance.

> **Please check the default settings!** By default, structure history that is both older than 30 days and less recent than 1,000 last changes (counted for each structure separately) will be removed daily. If you think that you need history of changes for a larger period, please review and make changes to the settings **immediately after the upgrade**!

Documentation: Automatic Structure Maintenance(see page 490)

7.23.2.5 Swapping panels in two-panel mode

When you have two structure panels open, you can now easily swap left and right panel by pressing **Alt+** keyboard shortcut.

7.23.2.6 JIRA 7.2 compatibility

Structure plugin and its extensions are now compatible with JIRA 7.2 and newer versions, including JIRA Data Center. The minimum supported version is JIRA 6.4.

7.23.2.7 Notable Fixes and Improvements

The following issues have been addressed in Structure 3.3:

- Added: Performance-related metrics, displayed to the JIRA admin, which would help ALM Works support diagnose performance-related issues faster.
- Added: Workaround for JIRA Data Center issues regarding cluster-based caches (JIRA issues JRA-62034[365] and JRA-62071[366]) that could cause JDC initialization failure.
- Fixed: Page freezes when removing extender in a structure with a lot of items.
- Fixed: Database troubleshooting page isn't accessible when database is locked.
- Fixed: Impossible to open issue page via clicking on the issue key on issue details panel.
- Fixed: Impossible to delete a structure that has slowly performing generators.
- Fixed: Slow activity stream queries.

7.23.3 Supported Versions

Structure 3.3 and all extensions support JIRA versions from 6.4 to 7.2. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported. JIRA Data Center is supported.

> Structure 3.3 is the last version of Structure to support JIRA 6. Structure 3.4 will most likely support JIRA versions starting from 7.0.

365 https://jira.atlassian.com/browse/JRA-62034
366 https://jira.atlassian.com/browse/JRA-62071

7.23.3.1 Structure.Pages Upgrade Required

Please note that if you're using Structure.Pages, you will have to upgrade it to version 1.1.0.

7.23.3.2 Structure.Testy Upgrade Required

Please note that if you're using Structure.Testy, you will have to upgrade it to version 2.1.0.

7.23.4 Installation and Upgrade

7.23.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.23.4.2 Upgrading Structure

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Upgrade Structure.Testy[367] and Structure.Pages[368] add-ons if you're using them.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.23.5 Structure 3.3.1 Release Notes

22nd of September 2016

Structure 3.3.1 is a patch release based on Structure 3.3.

367 https://marketplace.atlassian.com/plugins/com.almworks.testy
368 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[369]

7.23.5.1 Patch Release

> ⚠ If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has changed in Structure 3!

This is a patch release based on Structure 3.3. We have addressed the following issues:

- Fixed: exception during Structure restore from version 2.x.
- Fixed: dropdowns on Agile boards stop working after clicking on an issue key.
- Fixed: Structure disappears on an issue page after clicking on an issue key.
- Fixed: dropdowns stop working after issue page refresh.
- Fixed: exception in DefaultWorkflowSchemeManager.
- Fixed: incorrect initialization leading to "dangerous call" StructureRuntimeException.
- Fixed: slow loading of Agile boards based on structure() function.
- Fixed: warning in the logs about Inserter/Extender Duplicates Generator.

Upgrade is recommended for all Structure users.

7.23.5.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.23.5.3 Upgrade

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
> - If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

---

369 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.23.6 Structure 3.3.2 Release Notes

3rd of October 2016

Structure 3.3.2 is a patch release based on Structure 3.3.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[370]

7.23.6.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has been changed in Structure 3!

This is a patch release based on Structure 3.3. We have addressed a problem with issue key link not working on the Issue Navigator page in Internet Explorer 11.

Upgrade is recommended for all Structure users who use Internet Explorer.

7.23.6.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.23.6.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

370 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.23.7 Structure 3.3.3 Release Notes

15th of November 2016

Structure 3.3.3 is a patch release based on Structure 3.3.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[371]

7.23.7.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has been changed in Structure 3!

This is a patch release based on Structure 3.3.

We have addressed a potentially critical problem on JIRA Data Center: a node may be unable to perform re-indexing due to a lock held by Structure. This may happen if a node is added to or removed from a cluster.

Upgrade is recommended for all JIRA Data Center instances. This version supports JIRA 6.4 — 7.2.x.

7.23.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.23.7.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.

371 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.23.7.4 Enterprise Deployment Notes

We now include an additional section in Release Notes that aims to address the concerns of deploying the upgrade at a large enterprise. In this section we will suggest how the changes in the new version may affect stability and performance of JIRA and provide ideas for testing the new version on a staging environment.

In terms of stability and performance, this patch does not bring significant changes compared to versions 3.3.1 and 3.3.2. The only change was the fix for the reindex lockout problem, which was localized in the synchronization management code.

Additional testing and verification can be done on a staging environment if it is running Data Center as well. The possible testing scenario is:

- install a synchronizer on a test structure, verify that it is doing its job;
- add a node to the cluster, verify that reindex is possible on each node;
- locate the node that is running synchronization (use logs to see where the messages from synchronizers appear) and shut down that node;
- verify that the synchronizer continues running (it may take 5-10 minutes to switch to a new node) and that reindex is still possible on each node.

7.23.8 Structure 3.3.4 Release Notes

30th of January 2017

Structure 3.3.4 is a security patch release for JIRA 6.4.x.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[372]

7.23.8.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has been changed in Structure 3!

This is a patch release based on Structure 3.3.3. We have addressed two medium-to-critical security issues, affecting all Structure versions starting with 3.0. Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Upgrade is required for all JIRA instances running Structure versions 3.0–3.3.3 and using JIRA 6.4.x.

If you're using JIRA 7.x, please upgrade to Structure 3.5(see page 708) or later.

372 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

You should have Structure license with active maintenance (expiring not earlier than January 30th, 2017) to run the patches.

7.23.8.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.23.8.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.23.9 Structure 3.3.5 Release Notes

19th of June 2017

Structure 3.3.5 is a security patch release for JIRA 6.4.x.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[373]

7.23.9.1 Patch Release

This is a patch release based on Structure 3.3.4. We have addressed a critical security issue, affecting all Structure versions starting with 3.0. Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

373 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is required for all JIRA instances running Structure versions 3.0–3.3.4 and using JIRA 6.4.x.

If you're using JIRA 7.0.x., please upgrade to Structure 3.5.1(see page 714) or later version of Structure 3.5.x series.

If you're using JIRA 7.1 or later, please upgrade to Structure 4.1(see page 690) or later.

⚠ You should have Structure license with active maintenance (expiring not earlier than June 20th, 2017) to run the patches.

7.23.9.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may also want to check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.23.9.3 Upgrade

⚠
- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have Structure 3 beta or release candidate installed, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.24 Structure 3.2 Release Notes

ⓘ **18th of June, 2016**

Structure 3.2 adds separate backup and restore for Structure data, improves user experience on the project page and contains important fixes and optimizations.

⚠ **Important for Structure 2.x users!** If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 742) before upgrading.

Download the latest Structure and Extensions
(see page 978) Structure 3 Demo Server[374]
Structure 3 Quick Start Guide
(see page 732)

7.24.1 Version Highlights

Structure 3.2 is an incremental update in the Structure 3 series. It contains several new features, important improvements and bug fixes.

- Backup and restore Structure data
- Additional features on the project page
- Sequential index column
- Quick action lookup
- Structure gadget in Confluence
- Important fixes related to performance and stability

Upgrade is recommended; it is required if you experience any issues with the current version of Structure.

7.24.2 Changes in Detail

7.24.2.1 Backup and Restore

It's now possible to create a separate backup of the Structure data and later restore structures from backup. Although Structure data is also backed up when you make a full JIRA backup, the separate backup may be used to restore structures without affecting JIRA data or to export structures to other systems by reading the XML backup file.

We're still working on Migration feature, which will allow moving structures from one JIRA to another.

Documentation: Structure Backup, Restore and Migration(see page 485)

> Backup and Restore used to be Structure 2 features, but they were disabled with the release of Structure 3, as we've moved the storage from a separate database into the JIRA main database. Now they are being reinstated.

7.24.2.2 Structure on the Project Page

We have listened to our users who were unhappy with how Structure worked on the project page and we have made some improvements.

- When you open Structure tab on the project page, the displayed structure is automatically filtered to show you only the issues from the current project.
- A second panel is shown that displays all issues from the current project that are not added to the structure.

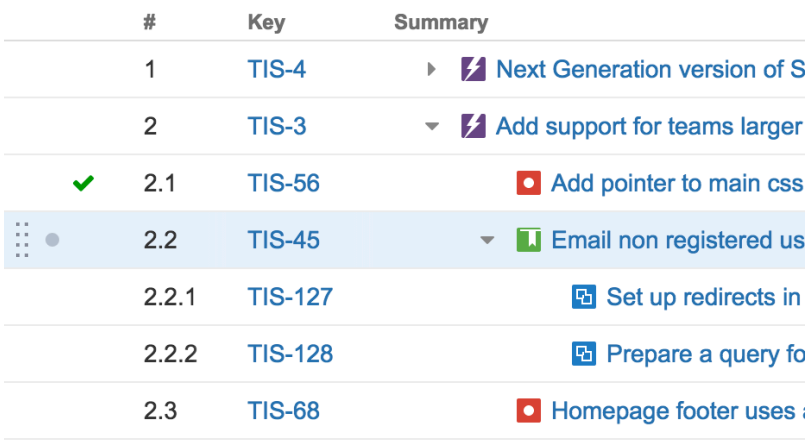
Documentation: Structure on the Project Page(see page 90)

374 https://demo-structure3.almworks.com/

7.24.2.3 Sequential Index Column

You can now add a column that would show you a hierarchical number (for example, "2.1.15"), based on the item's position in the hierarchy.

Documentation: Sequential Index Column(see page 318)



7.24.2.4 Quick Action Lookup

For those who love using keyboard, we added a special shortcut – **`s,q`** – that displays a lookup window where you can find all actions you can take by their name.

You can use this lookup to see if there's an action for what you need to do or to look up keyboard shortcuts for actions.

Documentation: Quick Action Lookup(see page 393)

7.24.2.5 Structure Gadget in Confluence

We have fixed several problems related to Structure's dashboard gadget being used in Confluence.

Unfortunately, a problem with loading custom fonts still remains due to an an issue in JIRA[375], so some of the custom Structure icons may not be displayed inside Confluence. There's a workaround – see Setting Up CORS Filter in JIRA(see page 542).

Documentation: Using Structure Gadget in Confluence(see page 732)

7.24.2.6 Notable Fixes and Other Improvements

The following issues have been addressed in Structure 3.2:

- Fixed: On JIRA Data Center, Health Check reports "com.almworks.jira.structure.autosync" cluster lock problem.
- Fixed: JIRA start may take a long time due to a cluster lock or database initialization issue.
- Fixed: Excessive warning messages in log files during reindex.
- Fixed: Performance issues for large structures.

375 https://jira.atlassian.com/browse/JRA-61400

7.24.3 Supported Versions

Structure 3.2 and all extensions support JIRA versions from 6.3 to 7.1. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

> ⚠ Structure 3.2 is the last minor version of Structure 3 to support JIRA 6.3. Structure 3.3 will most likely support JIRA versions starting from 6.4.

7.24.3.1 Structure.Pages Upgrade Required

Please note that if you're using Structure.Pages, you will have to upgrade it to version 1.0.1.

7.24.3.2 Structure.Testy Upgrade Required

Please note that if you're using Structure.Testy, you will have to upgrade it to version 2.0.2.

7.24.4 Installation and Upgrade

7.24.4.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.24.4.2 Upgrading Structure

> ⚠
> - If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
> - If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.x.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.

3. Upgrade Structure.Testy[376] and Structure.Pages[377] add-ons if you're using them.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.24.5 Structure 3.2.1 Release Notes

> **5th of July 2016**
>
> Structure 3.2.1 is a bugfix release based on Structure 3.2.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[378]

7.24.5.1 Patch Release

> If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has changed in Structure 3!

This is a patch release based on Structure 3.2. We have addressed the following issues:

- Fixed: cannot edit generators and folders after restore.
- Fixed: impossible to add new items after restore (Oracle, Postgres).
- Fixed: default view settings are not restored.
- Fixed: link extender doesn't react to new issues with links.
- Fixed: groupers with a level setting react incorrectly to generator and structure changes.
- Fixed: filters with a level setting remove everything on lower levels.
- Improvement: optimize synchronizer audit log table.

Upgrade is recommended to all Structure users.

7.24.5.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

376 https://marketplace.atlassian.com/plugins/com.almworks.testy
377 https://marketplace.atlassian.com/plugins/com.almworks.structure.pages/server/overview
378 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.24.5.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.24.6 Structure 3.2.2 Release Notes

30th of January 2017

Structure 3.2.2 is a security patch release for JIRA 6.3.x.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[379]

7.24.6.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has changed in Structure 3!

This is a patch release based on Structure 3.2.1. We have addressed two medium-to-critical security issues, affecting all Structure versions starting with 3.0. Details of the issues are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Upgrade is required for all JIRA instances running Structure versions 3.0–3.2.1 and using JIRA 6.3.x.

If you're using JIRA 6.4.x., please upgrade to Structure 3.3.4(see page 730) or later version of Structure 3.3.x series.

If you're using JIRA 7.x, please upgrade to Structure 3.5(see page 708) or later.

You should have Structure license with active maintenance (expiring not earlier than January 30th, 2017) to run the patches.

379 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.24.6.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.24.6.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.24.7 Structure 3.2.3 Release Notes

19th of June 2017

Structure 3.2.3 is a security patch release for JIRA 6.3.x.

Download from Archive(see page 979)
Structure on the Atlassian Marketplace[380]

7.24.7.1 Patch Release

This is a patch release based on Structure 3.2.2. We have addressed a critical security issue, affecting all Structure versions starting with 3.0. Details of the problem are not publicly disclosed at this time, but you can find more information in the restricted Structure Security google group: http://groups.google.com/group/structure-security-list (please mention your Structure SEN number or license ID when requesting access).

Upgrade is required for all JIRA instances running Structure versions 3.0–3.2.2 and using JIRA 6.3.x.

If you're using JIRA 6.4.x., please upgrade to Structure 3.3.5(see page 731) or later version of Structure 3.3.x series.

If you're using JIRA 7.0.x., please upgrade to Structure 3.5.1(see page 714) or later version of Structure 3.5.x series.

380 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

If you're using JIRA 7.1 or later, please upgrade to Structure 4.1(see page 690) or later.

⚠ You should have Structure license with active maintenance (expiring not earlier than June 20th, 2017) to run the patches.

7.24.7.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.24.7.3 Upgrade

⚠
- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.25 Structure 3.1 Release Notes

ⓘ **30th of April, 2016**

Structure 3.1 adds support for the upcoming Structure.Pages extension and contains a few fixes and improvements.

⚠ **Important for Structure 2.x users!** If you currently use Structure 2.11.2 or an earlier version, it is very important that you read Structure 3.0 Release Notes(see page 742) before upgrading.

Download Structure 3.0 and Extensions
(see page 978)Structure 3.0 Demo Server[381]

381 https://demo-structure3.almworks.com/

Structure 3 Quick Start Guide
(see page 739)

7.25.1 Incremental Update

Structure 3.1 is an incremental update in the Structure 3 series. It contains a number of fixes and improvements, and also adds support for the upcoming Structure.Pages release. There are no new features in this version.

Upgrade is recommended; it is required if you plan to install Structure.Pages or if you experience any issues with the current version of Structure.

7.25.1.1 Structure.Pages Release

Structure.Pages is our newest extension for Structure, which adds integration with Confluence and support for Confluence pages as another type of items in the structures. Check out Structure.Pages 1.0 Release Notes[382] for more details.

As this is the first public release of Structure.Pages, it will be some time until Structure.Pages appears at the Atlassian Marketplace. Until then, you can download Structure.Pages from the Download(see page 978) page.

7.25.1.2 Notable Fixes and Improvements

The following issues have been addressed in Structure 3.1:

- Added support for sorting by Script Runner's numeric custom fields.
- Improved Structure database lock-out during restore from backup.
- Fixed: Cannot upload changes after new issue creation fails.
- Fixed: Field column does not use proper number formatting for totals.
- Fixed: Excessive warnings from synchronization engine during restore from backup.
- Fixed: Changing direction in sorting by resolution doesn't work.
- Fixed: Error 500: "failed to upload structure changes" appears on creating issue under unavailable item

7.25.2 Supported Versions

Structure 3.1 and all extensions support JIRA versions from 6.3 to 7.1. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

Structure 3.1 is the last minor version of Structure 3 to support JIRA 6.3. Structure 3.2 will most likely support JIRA versions starting from 6.4.

7.25.2.1 Structure.Testy Upgrade Required

Please note that if you're using Structure.Testy, you will have to upgrade it to version 2.0.1.

382 https://wiki.almworks.com/display/pagesmaster/Structure.Pages+1.0+Release+Notes

7.25.3 Installation and Upgrade

7.25.3.1 Installing Structure

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.25.3.2 Upgrading Structure

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Upgrade Structure.Testy[383] add-on if you're using it.
4. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.25.4 Structure 3.1.1 Release Notes

2nd of June 2016

Structure 3.1.1 is a bugfix release based on Structure 3.1.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[384]

7.25.4.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has changed in Structure 3!

383 https://marketplace.atlassian.com/plugins/com.almworks.testy
384 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

This is a first patch release based on Structure 3.1. We have addressed some problems that were reported by our customers and made some improvements. Upgrade is recommended.

Among the issues addressed in Structure 3.1.1:

- Fixed: infinite generator creation in case of complicated structure.
- Fixed: a deadlock caused by JQL generators using the structure() function.
- Fixed: big syncronizer audit log retrieval.
- Implemented: synchronizer audit log cleanup.

7.25.4.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.25.4.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.26 Structure 3.0 Release Notes

5th of April, 2016

Structure 3.0 is the biggest, most extensive and most awaited update of the Structure add-on. It introduces a lot of amazing features that can help you bring your command of JIRA to the next level.

Important! Upgrade from Structure 2.x requires additional manual operations.

This version is a massive overhaul of Structure. If you're using Structure 2, please pay attention to these Release Notes, especially to Upgrade Instructions.

Download Structure 3.0 and Extensions
(see page 978)Structure 3.0 Demo Server[385]
Structure 3 Quick Start Guide
(see page 742)

7.26.1 Structure 3 – a Different Experience

Structure 3.0 is very different from previous versions of Structure. While the main concept – "structure" as a hierarchical list of things – remains, many things changed and a lot of features were added. We have put a lot of thought into expanding the scope of the product and rebuilt Structure almost from ground up.

After upgrading and finding your way around new user interface, you should be able to work with Structure 3.0 in the same way you worked with Structure 2.x. However, to take advantage of the new features such as Automation, some learning will be required. When you have Structure 3 installed, check out menu **Structure | Get Started** for a crash course on most important features. The Structure 3 Quick Start Guide(see page 742) has more details and is a recommended reading.

7.26.2 Structure 3 Highlights

- Multi-parent (same issue can be at multiple locations in any structure)
- Folders (special non-JIRA folders for containing issues in structures)
- Issue sorting and grouping
- Structures within structures
- Automation (automatic structure generation, an alternative to synchronizers)
- Revamped user interface
- JIRA Data Center compatible

Full description of all new features would be too much for this document. We invite you to take a look at the Structure 3 Quick Start Guide(see page 742) or explore the add-on yourself (make sure you stop by **Structure | Get Started** menu).

7.26.3 Notes on Structure 2 Features

⚠ Please read carefully: if you're relying on some of the changed features, you need to make sure that after upgrade everything works as expected.

7.26.3.1 Synchronizers

Synchronizers were remade. They generally do the same things as before, but we upgraded their logic so they can take advantage of the new architecture. For example, Links synchronizer can now place an issue in two or more places in the structure, so you will be able to visualize non-hierarchical links structure.

We kept most options, but introduced "Source of Truth", which helps bi-directional synchronizers resolve conflicts. If you're using synchronizers, please revisit all synchronization settings after upgrade.

385 https://demo-structure3.almworks.com

Synchronizers and Generators

Please note that generators "see" the results of synchronizers' work, but not vice versa. if you'd like to take advantage of the new feature, Automation(see page 122), and you're using synchronizers, we advise you to start off with a new structure to avoid confusion.

7.26.3.2 Other Changes

- Now an issue may be in several places in a structure. If you're using columns that calculate totals, the columns now have "Exclude duplicates" option, which makes sure that each issue is counted only once.
- Structure 3.0 does not have separate backup and restore, and there's no migration feature yet too. We'll reinstate these features in one of the upcoming versions.
- We changed the way Structure treats issues from projects that are not enabled for Structure. Such issues still won't be shown as a part of search result, however, if such an issue happens to be in a structure, it won't be hidden.
- If a user does not have access to issue A, but has access to its sub-issue B, the user will see "Unavailable item" instead of issue A. (In Structure 2, sub-issue B was elevated up one level to replace A in this case.)

7.26.4 Supported Versions

Structure 3.0 and all extensions support JIRA versions from 6.3 to 7.1. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

7.26.4.1 Browser Support

We now have only partial support for Internet Explorer 9 and 10. There are some known issues, but you may be able to use Structure. If you encounter a problem while using IE9 or IE10, please let us know and we'll advise if it's a known problem or something we can deal with.

If you're using Structure.Testy, you need to install Structure.Testy 2.0. Earlier versions of Testy are not supported.

7.26.5 Compatibility Issues

7.26.5.1 Other JIRA Plugins

If you're using a plugin that integrates with Structure 2.x, most likely it will not work with Structure 3.0. Here's the suggested course of action:

- Structure.Testy – we are releasing Structure.Testy 2.0, which is compatible with Structure 3.0. If you're using Testy, please upgrade Testy to the new version as well.
- `scheduled-sync`, `status-bar-column` or other complimentary plugins by ALM Works, published as a part of Structure 2.x documentation – these plugins will not work with Structure 3.0. We will release updated versions later, along with the Structure 3.0 final release. If you need one of those plugins urgently to try out Structure 3, please contact us.

- Custom plugins that integrate with Structure, whether developed by ALM Works or an independent vendor – these plugins will most likely not work with Structure 3.0 and will require additional work to be made compatible. For plugins made by ALM Works, please contact us with an upgrade request. If you have your own integrations or a plugin made by an independent contractor, you'll need information about the new APIs. Feel free to contact support@almworks.com[386] for this.
- Gantt-Chart for JIRA – if you're using this plugin and use its integration with Structure, you might want to hold off the upgrade until a compatible version is available.

7.26.5.2 REST API

REST APIs also have changed in Structure 3.0. If you have any reporting scripts that connect to Structure via REST, they most likely will stop working. At this moment, the new API is not published — please contact us at support@almworks.com[387] and we'll be happy to help you upgrade your scripts.

7.26.5.3 Remote Gadget Not Available

If you're using Structure dashboard gadget in Confluence, it is currently known to have issues. We'll be reviewing this functionality and possibly providing alternatives after Structure 3.0 release.

7.26.6 Changes in API

In Structure 3.0 we have made massive changes in the architecture of the product to accommodate the new features and lay groundwork for future expansion of the platform. Unfortunately, that means a lot of incompatible changes in the API. Most integrations with Structure 2.x will not work with Structure 3.0.

The new Java and REST API will be published later, closer to Structure 3.1 release. The reason for that is that we'll need to spend additional time after Structure 3.0 release to stabilize and clean the new APIs and provide sufficient documentation.

Until the new stable API is released, we can provide information about the new API on individual basis. Please feel free to contact us at support@almworks.com[388] if you'd like to integrate with Structure 3.0.

7.26.7 Installation and Upgrade

> **Important:** The data from Structure 2.x is not automatically transferred when you upgrade, so you'll need to manually migrate it after installation.

7.26.7.1 Installing Structure for the first time

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure 3.0 add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!

386 mailto:tosupport@almworks.com
387 mailto:support@almworks.com
388 mailto:support@almworks.com

2. When Add-on Manager reports about successful installation, click Get Startedto visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.26.7.2 Upgrading from Structure 2.9–2.11

If you have Structure version older than 2.9, please upgrade to Structure 2.11.2 version first.

1. Create backup of current Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the whole `structure/` sub-directory under JIRA home while Structure plugin is disabled.
2. Download and install Structure 3.0 add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
3. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
4. To transfer the data, open **Administration | Structure | Export Structure 2.x Data** page. It allows you to create a backup zip with all Structure 2.x data, and then opens **Restore Structure** page, allowing you to immediately import the backup into Structure 3.x database.

   Important: after you press **Create Backup**, wait for the backup to finish – it might take a while! Do not try to go back and repeat the backup. To see if the backup is still going on, access JIRA home directory, `export/` folder, and see if the backup file is growing in size.

   a. Alternatively, you can use **Administration | Structure | Restore Structure** menu and use any Structure 2.x backup made earlier.
5. If you have Structure.Testy installed, upgrade to Structure.Testy version 2.0 or later.
6. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Unlike previous versions, Structure 3.0 uses the main JIRA database to store its data. After upgrade, your Structure 3.0 database will be empty. Don't panic – all data is intact, but it must be transferred from Structure 2.x database. See the instructions above.

7.26.7.3 Upgrading from preview versions of Structure 3.0

If you have installed Structure 3 beta or release candidate, you need to uninstall it first.

1. Download and install Structure 3.0 add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version! All data will be safe.
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. If you have Structure.Testy installed, upgrade to Structure.Testy version 2.0 or later.

4. If you have a preview of Structure.Pages installed, please contact support@almworks.com[389] for a link to Structure.Pages beta3 version. Earlier versions will not work.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.26.7.4 Upgrading "Global Structure"

If you're using "Global Structure" structure, which was created by default in Structure 2.x, you need to make sure that there's an "owner" of that structure. Otherwise, Automation feature will not work there.

1. Open **Structure | Manage Structure**.
2. Find Global Structure and check if it has non-empty Owner.
3. If it doesn't have an owner, click **Configure**, and set yourself as the owner.

7.26.7.5 Downgrading

If you decide to downgrade to Structure 2.11, you can do so, but any changes made in Structure 3.0 will not be transferred to the previous version.

7.26.8 Feedback Is Welcome!

Please let us know what you think about the new Structure! We have grandiose plans for next versions, which now will be coming out frequently, and your feedback is instrumental for aligning our plans with what the users need.

Please write to support@almworks.com[390] or almworks twitter[391] if you have any comments, questions, suggestions or feelings to share!

7.26.9 Structure 3.0.1 Release Notes

> **11th of April 2016**
>
> Structure 3.0.1 has critical fixes related to SQL Server database, upgrade from Structure 2, performance issues and compatibility with JIRA 7.1.4.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[392]

7.26.9.1 Patch Release

> If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has changed in Structure 3!

This is a first patch release based on Structure 3.0. We have addressed some critical problems that have surfaced after the initial Structure 3. Upgrade is highly recommended.

389 mailto:support@almworks.com
390 mailto:support@almworks.com
391 https://twitter.com/almworks
392 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

SQL Server Issues

There were a number of problems that appeared on some JIRA instances that were running on SQL Server database. Most prominently, there was a chance that the add-on would not work with a "*Invalid object name*" message.

Additionally, there were potential problems and database connection leaks when restoring Structure data from backup. All these are fixed in Structure 3.0.1.

> If you have been using Structure 3.0 and have migrated your data from Structure 2, please check your Database Monitoring page when there's no activity in JIRA. If you see a constant number of active connections in the pool when there's no work load, consider restarting JIRA.

JIRA 7.1.4 Compatibility

Some backwards-incompatible changes were introduced in Atlassian JIRA 7.1.4, resulting in the layout switch not working.

Structure 3.0.1 is fully compatible with JIRA 6.3 – 7.1.4.

Improved Migration

Unfortunately, we didn't attract enough attention to the fact that when you upgrade to Structure 3 from Structure 2, you need to re-import all of your Structure data. This introduced some confusion, as the structures appeared to be lost.

To improve the migration process, we now offer JIRA administrator to upgrade Structure data immediately, using a top-of-the-screen banner.

Additionally, the backup and restore processes now happen in the background, so the browsers won't time out when the procedure takes longer than usual.

> **Important**! If you already have migrated your Structure 2.x data, just close the banner that suggests you to do the migration (again). If you restore Structure 2.x data once more, you will roll back your Structure data to the state before the upgrade.

Miscellaneous Fixes and Improvements

- Removed status icons on JIRA 7+
- Performance optimizations
- Smaller fixes and improvements

7.26.9.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.

3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.26.9.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from version 3.0.0 is simple:

1. Create backup of JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data.
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.26.10 Structure 3.0.2 Release Notes

26th of April 2016

Structure 3.0.2 is a bugfix release based on Structure 3.0.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[393]

7.26.10.1 Patch Release

If you are using Structure 2, we encourage you to read Structure 3.0 Release Notes(see page 742) – a lot has changed in Structure 3!

This is a second patch release based on Structure 3.0. We have addressed some problems that were reported by our customers and made some improvements. Upgrade is recommended.

Among the issues addressed in Structure 3.0.2:

- Fixed: synchronizers cycle protection fails to send e-mails to structure owners.
- Fixed: synchronizers cycle protection causes synchronizers become instantly disabled if a cycle happens at least once.
- Fixed: problems with reassigning issues via drag-and-drop when the assignee has their login name changed in the past, or when the user is imported from an LDAP directory. (Also affects other fields that refer to users.)
- Fixed: structure synchronizers created in Structure plugin version 1.3 or earlier are not properly migrated during transition from Structure 2 to Structure 3.
- Fixed: printable page and Excel export do not show values for certain fields, such as Total Remaining Estimate and other duration-type fields, Total Votes and various configurations of the Progress field.

393 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

- Fixed: error when trying to create a new structure with Agile template, but the Agile board does not have Rank ordering.
- Fixed: printable page and Excel export disregard "Exclude Duplicates" setting for the column.

7.26.10.2 Installation

If your JIRA server does not have Structure yet, the installation is simple:

1. Download and install Structure add-on, either from Atlassian Marketplace or from Download(see page 978) page. Pick the correct version based on your JIRA version!
2. When Add-on Manager reports about successful installation, click Get Started to visit a page with important guidance for the JIRA administrator. You may want to also check out the user's Get Started page, available under "Structure" top-level menu.
3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.26.10.3 Upgrade

- If you're upgrading from version 2.11.2 or earlier, please read Structure 3.0.0 Release Notes(see page 742).
- If you have installed Structure 3 beta or release candidate, please read Structure 3.0.0 Release Notes(see page 742).

Upgrade procedure from versions 3.0.x is simple:

1. Consider backing up JIRA data. Use **Administration | System | Backup System**. Starting from version 3.0.0 Structure data can be backed up together with JIRA data. (If you have a large instance and have proper backup strategy in place, you may skip this step.)
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.26.11 Preview Version Release Notes

- Structure 3.0 RC1 Release Notes(see page 750)
- Structure 3.0 Beta 2 Release Notes(see page 755)
- Structure 3.0 Beta 1 Release Notes(see page 759)

7.26.11.1 Structure 3.0 RC1 Release Notes

These Release Notes are mostly a copy of the previous Release Notes for Structure 3.0 beta 2. Changes and new information are marked with **RC1**.

23 March 2016

Structure 3.0 is going to be a massive update and the most important of all Structure updates. This is the first release candidate of Structure 3.0.

Download Structure 3.0 and Extensions(see page 978)
Structure 3.0 Demo Server[394]
Structure 3 Quick Start Guide(see page 750)
Structure.Pages Quick Start Guide(see page 750)

Structure 3 – a Different Experience

Structure 3.0 is very different from previous versions of Structure. While the main concept – "structure" as a hierarchical list of things – remains, many things changed and a lot of features were added. We have put a lot of thought into expanding the scope of the product and rebuilt Structure almost from the ground up.

After upgrading and finding your way around new user interface, you should be able to work with Structure 3.0 in the same way you worked with Structure 2.x. However, to take advantage of the new features such as Automation, some learning will be required. The Structure 3 Quick Start Guide(see page 750) is a great way to get started with the new features.

Structure 3 Highlights

- Structure.Pages – a new Structure extension
- JIRA Data Center compatible
- Multi-parent (same issue can be at multiple locations in any structure)
- Folders (special non-JIRA folders for containing issues in structures)
- Issue sorting and grouping
- Structures within structures
- Automation (Automatic structure generation, an alternative to synchronizers)
- Revamped user interface

Take a look at the Structure 3 Quick Start Guide(see page 750) for more information about the new features.

What's New in RC 1 RC1

New Structure Wizard

With this version, we have added a wizard that can help you build an initial structure in a few simple steps, using one of the templates. The wizard appears when you select **Structure | Create Structure** menu.

We believe it will help new users get started with the powerful Automation feature.

Synchronization Audit Log

Now all changes that synchronizers apply to the JIRA issues are logged in the database, and Structure allows JIRA administrator to search / browse the changes made. In case some of the changes were not desired, it is possible to undo them.

The Audit Log is available in **Administration | Structure | Support** menu**.**

Note that the synchronizers must log the changes they make through Structure API, otherwise they won't appear in audit log. All standard synchronizers do that, but a custom synchronizer may be written without logging JIRA changes.

394 https://demo-structure3.almworks.com

Updated Keyboard Shortcuts

We have reviewed and updated keyboard shortcuts. Check out the new shortcuts by pressing *Ctrl+?.*

History Migration

Structure history is now restored when you migrate data from Structure 2.x backup.

Fixes and Performance Improvements

There were a lot of bug fixes and improvements in performance and stability since beta2.

Supported Versions

Structure 3.0 and all extensions support JIRA versions from 6.3 to 7.1. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

Structure.Pages supports Confluence versions from 5.7 to 5.9.

> ⚠ If you're using Structure.Testy, you need to install Structure.Testy 2.0.0.rc1 with Structure 3.0.0.rc1. Earlier versions of Testy are not supported.

Compatibility Issues

Other JIRA Plugins

If you’re using a plugin that integrates with Structure 2.x, most likely it will not work with Structure 3.0. Here’s the suggested course of action:

- Structure.Testy – we are releasing Structure.Testy 2.0.0.rc1 that is compatible with Structure 3.0. If you’re using Testy, please upgrade Testy to the new version as well.
- `scheduled-sync`, `status-bar-column` or other complimentary plugins by ALM Works, published as a part of Structure 2.x documentation – these plugins will not work with Structure 3.0. We will release updated versions later, along with the Structure 3.0 final release. If you need one of those plugins urgently to try out Structure 3, please contact us.
- Custom plugins that integrate with Structure, whether developed by ALM Works or an independent vendor – these plugins will most likely not work with Structure 3.0 and will require additional work to be made compatible. For plugins made by ALM Works, please contact us with an upgrade request. If you have your own integrations or a plugin made by an independent contractor, you’ll need information about the new APIs. Feel free to contact support@almworks.com[395] for this.
- Gantt-Chart for JIRA – if you’re using this plugin from Frank Polscheit and use its integration with Structure, you might want to hold off the upgrade until a compatible version is available.

395 mailto:tosupport@almworks.com

REST API

REST APIs also have changed in Structure 3.0. If you have any reporting scripts that connect to Structure via REST, they most likely will stop working. At this moment, the new API is not published — please contact us at support@almworks.com[396] and we'll be happy to help you upgrade your scripts.

Remote Gadget Not Available

If you're using Structure dashboard gadget in Confluence, it is currently known to have issues. We'll be reviewing this functionality and possibly providing alternatives after Structure 3.0 release.

Changes in API

In Structure 3.0 we have made massive changes in the architecture of the product to accommodate the new features and lay groundwork for future expansion of the platform. Unfortunately, that means a lot of incompatible changes in the API. Most integrations with Structure 2.x will not work with Structure 3.0.

The new Java and REST API will be published later, closer to Structure 3.1 release. The reason for that is that we'll need to spend additional time after Structure 3.0 release to stabilize and clean the new APIs and provide sufficient documentation.

Until the new stable API is released, we can provide information about the new API on individual basis. Please feel free to contact us at support@almworks.com[397] if you'd like to integrate with Structure 3.0.

Documentation

At this time, the documentation for Structure 3.0 is in the works. We'll release full documentation along with the final release of Structure 3. Please refer to the Structure 3 Quick Start Guide(see page 750) for a short introduction to Structure 3.

Installation and Upgrade

This version or some of the extensions may not published on the Atlassian Marketplace, you'll need to manually download and install it. Also, the data from Structure 2.x is not automatically transferred, so you'll need to manually upgrade it.

The download links are available at Download Structure 3.0(see page 978) page. **RC1**

Installing Structure for the first time

If your JIRA server does not have Structure yet, you can install Structure using these steps:

1. Download Structure 3.0. Pick the correct version based on your JIRA version!
2. Open **Administration | Add-ons | Manage Add-ons** and use **Upload add-on** link to install the downloaded plugin into your JIRA.
3. If you'd like to install Structure.Testy, download and install it in the same manner.
4. If you'd like to install Structure.Pages, download and install it in JIRA, and also you need to install a special system plugin in Confluence. Please refer to Structure.Pages Quick Start Guide(see page 750).
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

---

396 mailto:support@almworks.com
397 mailto:support@almworks.com

Upgrading from Structure 2.9–2.11

If you have Structure version older than 2.9, please upgrade to the latest Structure 2.x version first.

1. Create backup of current Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the whole `structure/` sub-directory under JIRA home while Structure plugin is disabled. See Backing Up Structure(see page 486) for details.
2. Download Structure 3.0. Pick the correct version based on your JIRA version!
3. Open **Administration | Add-ons | Manage Add-ons** and use **Upload add-on** link to install the downloaded plugin into your JIRA.
4. To transfer the data, open **Administration | Structure | Export Structure 2.x Data** page. It allows you to create a backup zip with all Structure 2.x data, and then opens **Restore Structure** page, allowing you to immediately import the backup into Structure 3.x database.
    a. Alternatively, you can use **Administration | Structure | Restore Structure** menu and use any Structure 2.x backup made earlier.
5. If you have Structure.Testy installed, download and install the latest beta version of Structure.Testy.
6. If you'd like to install Structure.Pages, download and install it in JIRA, and also you need to install a special system plugin in Confluence. Please refer to Structure.Pages Quick Start Guide(see page 750).
7. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Unlike previous versions, Structure 3.0 uses the main JIRA database to store its data. After upgrade, your Structure 3.0 database will be empty. Don't panic – all data is intact, but it must be transferred from Structure 2.x database. See the instructions above.

Upgrading from previous versions of Structure 3.0

If you have installed Structure 3 beta or a preview version, you need to uninstall it first.

1. Download Structure 3.0. Pick the correct version based on your JIRA version!
2. Open **Administration | Add-ons | Manage Add-ons** and uninstall your current Structure version. All data will be safe.
3. Use **Upload add-on** link to install the downloaded plugin into your JIRA.
4. If you'd like to install Structure.Testy, download and install it. Structure.Testy 2.0.0.rc1 is the earliest version compatible with Structure 3.0.0.rc1.
5. If you'd like to install Structure.Pages, download and install it in JIRA, and also you need to install a special system plugin in Confluence. Please refer to Structure.Pages Quick Start Guide(see page 750).
6. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading "Global Structure" RC1

If you're using "Global Structure" structure, which was created by default in Structure 2.x, you need to make sure that there's an "owner" of that structure. Otherwise, Automation feature will not work there.

1. Open **Structure | Manage Structure**.
2. Find Global Structure and check if it has non-empty Owner.
3. If it doesn't have an owner, click **Configure**, and set yourself as the owner.

Downgrading

If you decide to downgrade to Structure 2.11, you can do so, but any changes made in Structure 3.0 will not be transferred to the previous version.

Feedback Is Welcome!

Please let us know what you think about the new Structure! We value your feedback very much and we still have time to make some final changes before official Structure 3.0 release.

Please write to support@almworks.com[398] or almworks twitter[399] if you have any comments, questions, suggestions or feelings to share!

7.26.11.2 Structure 3.0 Beta 2 Release Notes

> **3 March 2016**
>
> Structure 3.0 is going to be a massive update and the most important of all Structure updates. This is the second beta version of Structure 3.0.

Subscribe and Download Structure 3.0.0.beta2[400] (you will also receive a link to Structure.Testy 2.0.0.beta2 and Structure.Pages 1.0.0.beta2)
Structure 3.0 Demo Server[401]

Structure 3 – a Different Experience

Structure 3.0 is very different from previous versions of Structure. While the main concept – "structure" as a hierarchical list of things – remains, many things changed and a lot of features were added. We have put a lot of thought into expanding the scope of the product and rebuilt Structure almost from the ground up.

After upgrading and finding your way around new user interface, you should be able to work with Structure 3.0 in the same way you worked with Structure 2.x. However, to take advantage of the new features such as Automation, some learning will be required. The Structure 3 Quick Start Guide(see page 755) is a great way to get started with the new features.

Structure 3 Highlights

- Structure.Pages – a new Structure extension (**available with Beta 2**)
- JIRA Data Center compatible
- Multi-parent (same issue can be at multiple locations in any structure)
- Folders (special non-JIRA folders for containing issues in structures)
- Issue sorting and grouping
- Structures within structures
- Automation (Automatic structure generation, an alternative to synchronizers)
- Revamped user interface

Take a look at the Structure 3 Quick Start Guide(see page 755) for more information about the new features.

---

398 mailto:support@almworks.com
399 https://twitter.com/almworks
400 http://almworks.com/s3/#beta
401 https://demo-structure3.almworks.com

What's New in Beta 2

Structure 3.0.0.beta2 includes several fixes and improvements. It also comes with beta2 versions of our Structure extensions.

Structure.Pages

With this version, we're publishing our new plugin, Structure.Pages. It is a Structure extension that lets you bring together content from Confluence and JIRA. We're very excited about this new product and invite you to try it.

Structure.Pages Quick Start Guide(see page 755) will help you install the plugin and understand how to use it.

Structure.Testy Update

We're also releasing Structure.Testy 2.0.0.beta2, which has compatibility fixes for JIRA 7.1.

Supported Versions

Structure 3.0 and all extensions support JIRA versions from 6.3 to 7.1. Note that for Structure there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

Structure.Pages supports Confluence versions from 5.7 to 5.9.

If you're using Structure.Testy, you need to install Structure.Testy 2.0.0.beta2 with Structure 3.0.0.beta2. Earlier versions of Testy are not supported.

Compatibility Issues

Other JIRA Plugins

If you’re using a plugin that integrates with Structure 2.x, most likely it will not work with Structure 3.0. Here’s the suggested course of action:

- Structure.Testy – we are releasing Structure.Testy 2.0.0.beta2 that is compatible with Structure 3.0. If you’re using Testy, please upgrade Testy to the new version as well.
- `scheduled-sync`, `status-bar-column` or other complimentary plugins by ALM Works, published as a part of Structure 2.x documentation – these plugins will not work with Structure 3.0. We will release updated versions later, along with the Structure 3.0 final release. If you need one of those plugins urgently to try out the Beta, please contact us.
- Custom plugins that integrate with Structure, whether developed by ALM Works or an independent vendor – these plugins will most likely not work with Structure 3.0 and will require additional work to be made compatible. For plugins made by ALM Works, please contact us with an upgrade request. If you have your own integrations or a plugin made by an independent contractor, you’ll need information about the new APIs. Feel free to contact support@almworks.com[402] for this.
- Gantt-Chart for JIRA – if you’re using this plugin from Frank Polscheit and use its integration with Structure, you might want to hold off the upgrade until a compatible version is available.

402 mailto:tosupport@almworks.com

REST API

REST APIs also have changed in Structure 3.0. If you have any reporting scripts that connect to Structure via REST, they most likely will stop working. At this moment, the new API is not published — please contact us at support@almworks.com[403] and we'll be happy to help you upgrade your scripts.

Remote Gadget Not Available

If you're using Structure dashboard gadget in Confluence, it is currently known to have issues. We'll be reviewing this functionality and possibly providing alternatives after Structure 3.0 release.

Changes in API

In Structure 3.0 we have made massive changes in the architecture of the product to accommodate the new features and lay groundwork for future expansion of the platform. Unfortunately, that means a lot of incompatible changes in the API. Most integrations with Structure 2.x will not work with Structure 3.0.

The new Java and REST API will be published later, closer to Structure 3.1 release. The reason for that is that we'll need to spend additional time after Structure 3.0 release to stabilize and clean the new APIs and provide sufficient documentation.

Until the new stable API is released, we can provide information about the new API on individual basis. Please feel free to contact us at support@almworks.com[404] if you'd like to integrate with Structure 3.0.

Documentation

At this time, the documentation for Structure 3.0 is in the works. We'll release full documentation along with the final release of Structure 3. Please refer to the Structure 3 Quick Start Guide(see page 755) for a short introduction to Structure 3.

Installation and Upgrade

This version is not published on the Atlassian Marketplace, you'll need to manually download and install it. Also, the data from Structure 2.x is not automatically transferred, so you'll need to manually upgrade it.

> ⚠ Download links for beta versions of Structure, Structure.Testy and Structure.Pages are distributed by email. Subscribe at Structure 3.0 Home Page[405] and you'll receive the links in a couple of minutes. If you have already subscribed, but didn't get the email with the links to **beta2** versions of the products, please contact info@almworks.com[406].

Installing Structure for the first time

If your JIRA server does not have Structure yet, you can install Structure using these steps:

1. Download Structure 3.0 Beta. Pick the correct version based on your JIRA version!

403 mailto:support@almworks.com
404 mailto:support@almworks.com
405 http://almworks.com/s3/#beta
406 mailto:info@almworks.com

2. Open **Administration | Add-ons | Manage Add-ons** and use **Upload add-on** link to install the downloaded plugin into your JIRA.
3. If you'd like to install Structure.Testy, download and install it in the same manner.
4. If you'd like to install Structure.Pages, download and install it in JIRA, and also you need to install a special system plugin in Confluence. Please refer to Structure.Pages Quick Start Guide(see page 755).
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Upgrading from Structure 2.9–2.11

If you have Structure version older than 2.9, please upgrade to the latest Structure 2.x version first.

1. Create backup of current Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the whole `structure/` sub-directory under JIRA home while Structure plugin is disabled. See Backing Up Structure(see page 486) for details.
2. Download Structure 3.0 Beta. Pick the correct version based on your JIRA version!
3. Open **Administration | Add-ons | Manage Add-ons** and use **Upload add-on** link to install the downloaded plugin into your JIRA.
4. To transfer the data, open **Administration | Structure | Export Structure 2.x Data** page. It allows you to create a backup zip with all Structure 2.x data, and then opens **Restore Structure** page, allowing you to immediately import the backup into Structure 3.x database.
    a. Alternatively, you can use **Administration | Structure | Restore Structure** menu and use any Structure 2.x backup made earlier.
5. If you have Structure.Testy installed, download and install the latest beta version of Structure.Testy.
6. If you'd like to install Structure.Pages, download and install it in JIRA, and also you need to install a special system plugin in Confluence. Please refer to Structure.Pages Quick Start Guide(see page 755).
7. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Unlike previous versions, Structure 3.0 uses the main JIRA database to store its data. After upgrade, your Structure 3.0 database will be empty. Don't panic – all data is intact, but it must be transferred from Structure 2.x database. See the instructions above.

Upgrading from previous versions of Structure 3.0

If you have installed Structure 3.0.0.beta1 or a preview version of Structure 3.0, you need to uninstall it first.

1. Download Structure 3.0 Beta. Pick the correct version based on your JIRA version!
2. Open **Administration | Add-ons | Manage Add-ons** and uninstall your current Structure version. All data will be safe.
3. Use **Upload add-on** link to install the downloaded plugin into your JIRA.
4. If you'd like to install Structure.Testy, download and install it. Structure.Testy 2.0.0.beta2 is the earliest version compatible with Structure 3.0.0.beta2.
5. If you'd like to install Structure.Pages, download and install it in JIRA, and also you need to install a special system plugin in Confluence. Please refer to Structure.Pages Quick Start Guide(see page 755).
6. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Downgrading

If you decide to downgrade to Structure 2.11, you can do so, but any changes made in Structure 3.0 will not be transferred to the previous version.

Feedback Is Welcome!

Please let us know what you think about the new Structure! We value your feedback very much and we still have time to make some final changes before official Structure 3.0 release.

Please write to support@almworks.com[407] or almworks twitter[408] if you have any comments, questions, suggestions or feelings to share!

7.26.11.3 Structure 3.0 Beta 1 Release Notes

> **15 February 2016**
>
> Structure 3.0 is going to be a massive update and the most important of all Structure updates. Today we release the first publicly available beta version of Structure 3.0.

Subscribe and Download Structure 3.0.0.beta1[409] (you will also receive a link to Structure.Testy 2.0.0.beta1)
Structure 3.0 Demo Server[410]

A Different Experience

Structure 3.0 is very different from previous versions of Structure. While the main concept – "structure" as a hierarchical list of things – remains, many things changed and a lot of features were added. We have put a lot of thought into expanding the scope of the product and rebuilt Structure almost from the ground up.

After upgrading and finding your way around new user interface, you should be able to work with Structure 3.0 in the same way you worked with Structure 2.x. However, to take advantage of the new features such as Automation, some learning will be required. The Structure 3 Quick Start Guide(see page 759) is a great way to get started with the new features.

Highlights

- JIRA Data Center compatible
- Multi-parent (same issue can be at multiple locations in any structure)
- Folders (special non-JIRA folders for containing issues in structures)
- Issue sorting and grouping
- Structures within structures
- Automation (Automatic structure generation, an alternative to synchronizers)
- Revamped user interface

Take a look at the Structure 3 Quick Start Guide(see page 759) for more information about the new features.

Supported JIRA Versions

Structure 3.0 supports JIRA versions from 6.3 to 7.1. Note that there are separate downloadable files for JIRA 6 and JIRA 7. All editions of JIRA (JIRA Core, JIRA Software, JIRA Service Desk) are supported.

---

407 mailto:support@almworks.com
408 https://twitter.com/almworks
409 http://almworks.com/s3/#beta
410 https://demo-structure3.almworks.com

Compatibility Issues

Other JIRA Plugins

If you're using a plugin that integrates with Structure 2.x, most likely it will not work with Structure 3.0. Here's the suggested course of action:

- Structure.Testy – we are releasing Structure.Testy 2.0.0.beta1 that is compatible with Structure 3.0. If you're using Testy, please upgrade Testy to the new version as well.
- `scheduled-sync`, `status-bar-column` or other complimentary plugins by ALM Works, published as a part of Structure 2.x documentation – these plugins will not work with Structure 3.0. We will release updated versions later, along with the Structure 3.0 final release. If you need one of those plugins urgently to try out the Beta, please contact us.
- Custom plugins that integrate with Structure, whether developed by ALM Works or an independent vendor – these plugins will most likely not work with Structure 3.0 and will require additional work to be made compatible. For plugins made by ALM Works, please contact us with an upgrade request. If you have your own integrations or a plugin made by an independent contractor, you'll need information about the new APIs. Feel free to contact support@almworks.com[411] for this.
- Gantt-Chart for JIRA – if you're using this plugin from Frank Polscheit and use its integration with Structure, you might want to hold off the upgrade until a compatible version is available.

REST API

REST APIs also have changed in Structure 3.0. If you have any reporting scripts that connect to Structure via REST, they most likely will stop working. At this moment, the new API is not published — please contact us at support@almworks.com[412] and we'll be happy to help you upgrade your scripts.

Changes in API

In Structure 3.0 we have made massive changes in the architecture of the product to accommodate the new features and lay groundwork for future expansion of the platform. Unfortunately, that means a lot of incompatible changes in the API. Most integrations with Structure 2.x will not work with Structure 3.0.

The new Java and REST API will be published later, closer to Structure 3.1 release. The reason for that is that we'll need to spend additional time after Structure 3.0 release to stabilize and clean the new APIs and provide sufficient documentation.

Until the new stable API is released, we can provide information about the new API on individual basis. Please feel free to contact us at support@almworks.com[413] if you'd like to integrate with Structure 3.0.

Documentation

At this time, the documentation for Structure 3.0 is in the works. We'll release full documentation along with the final release of Structure 3. Please refer to the Structure 3 Quick Start Guide(see page 759) for a short introduction to Structure 3.

411 mailto:tosupport@almworks.com
412 mailto:support@almworks.com
413 mailto:support@almworks.com

Installation and Upgrade

This version is not published on the Atlassian Marketplace, you'll need to manually download and install it. Also, the data from Structure 2.x is not automatically transferred, so you'll need to manually upgrade it.

1. Create backup of current Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the whole `structure/` sub-directory under JIRA home while Structure plugin is disabled. See Backing Up Structure(see page 486) for details.
2. Download Structure 3.0 Beta (subscribe at the Structure 3.0 Home Page[414] and you'll receive the links by email). Pick the correct version based on your JIRA version!
3. Open **Administration | Add-ons | Manage Add-ons** and use **Upload add-on** link to install the downloaded plugin into your JIRA.
    a. If you have an early preview version of Structure 3.0 installed, uninstall it first.
4. To transfer the data, open **Administration | Structure | Export Structure 2.x Data** page. It allows you to create a backup zip with all Structure 2.x data, and then opens **Restore Structure** page, allowing you to immediately import the backup into Structure 3.x database.
    a. Alternatively, you can use **Administration | Structure | Restore Structure** menu and use any Structure 2.x backup made earlier.
5. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

Unlike previous versions, Structure 3.0 uses the main JIRA database to store its data. After upgrade, your Structure 3.0 database will be empty. Don't panic – all data is intact, but it must be transferred from Structure 2.x database. See the instructions above.

Downgrading

If you decide to downgrade to an earlier version of Structure, you can do so, but any changes made in Structure 3.0 will not be transferred to the previous version.

Feedback Is Welcome

Please let us know what you think about the new Structure! We value your feedback very much and we still have time to make some final changes before official Structure 3.0 release.

Please write to support@almworks.com[415] or almworks twitter[416] if you have any comments, questions, suggestions or feelings to share!

7.27 (HIDDEN) Release Notes (Structure 1-2)

Pending delete

All release notes:

414 http://almworks.com/s3/#beta
415 mailto:support@almworks.com
416 https://twitter.com/almworks

- Structure 2.9 Release Notes(see page 763)
  - API Changes in Structure 2.9(see page 765)
- Structure 2.8 Release Notes(see page 766)
  - Structure 2.8.3 Release Notes(see page 770)
  - Structure 2.8.2 Release Notes(see page 769)
  - Structure 2.8.1 Release Notes(see page 768)
  - API Changes in Structure 2.8(see page 770)
- Structure 2.7 Release Notes(see page 771)
  - API Changes in Structure 2.7(see page 773)
- Structure 2.6 Release Notes(see page 774)
  - Structure 2.6.1 Release Notes(see page 776)
  - API Changes in Structure 2.6(see page 777)
- Structure 2.5 Release Notes(see page 777)
  - Structure 2.5.3 Release Notes(see page 783)
  - Structure 2.5.2 Release Notes(see page 782)
  - Structure 2.5.1 Release Notes(see page 781)
  - API Changes in Structure 2.5(see page 784)
- Structure 2.4 Release Notes(see page 785)
  - Structure 2.4.3 Release Notes(see page 790)
  - Structure 2.4.2 Release Notes(see page 789)
  - Structure 2.4.1 Release Notes(see page 787)
  - API Changes in Structure 2.4(see page 791)
- Structure 2.3 Release Notes(see page 792)
  - API Changes in Structure 2.3(see page 793)
- Structure 2.2 Release Notes(see page 794)
  - Structure 2.2.1 Release Notes(see page 796)
- Structure 2.1 Release Notes(see page 797)
  - Structure 2.1.1 Release Notes(see page 800)
  - API Changes in Structure 2.1(see page 801)
- Structure 2.0 Release Notes(see page 801)
  - API Changes in Structure 2.0(see page 809)
- Structure 1.7 Release Notes(see page 810)
  - Structure 1.7.1 Release Notes(see page 812)
  - API Changes in Structure 1.7(see page 813)
- Structure 1.6 Release Notes(see page 813)
  - API Changes in Structure 1.6(see page 817)
- Structure 1.5 Release Notes(see page 819)
  - Structure 1.5.2 Release Notes(see page 821)
  - Structure 1.5.1 Release Notes(see page 821)
  - API Changes in Structure 1.5(see page 822)
- Structure 1.4 Release Notes(see page 823)
  - Structure 1.4.1 Release Notes(see page 826)
  - API Changes in Structure 1.4(see page 827)
- Structure 1.3 Release Notes(see page 828)
  - Structure 1.3.2 Release Notes(see page 831)
  - Structure 1.3.1 Release Notes(see page 830)
- Structure 1.2 Release Notes(see page 832)
- Structure 1.1 Release Notes(see page 836)
  - Structure 1.1.1 Release Notes(see page 838)
- Structure 1.0 Release Notes(see page 839)

7.27.1 Structure 2.9 Release Notes

8 September 2014

Structure 2.9 shows issue details on the Structure Board, adds the ability to archive old structures, provides a new column that can aggregate work logged over a specific time period, and contains a lot of other important improvements and bug fixes.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[417]

7.27.1.1 Version Highlights

- Viewing full issue details without leaving Structure Board
- A column that shows work logged over a period
- Visual notification after issues are removed from structure
- Archiving old structures
- JIRA administrators can change the owner of a structure
- Copying a structure including installed synchronizers

7.27.1.2 Changes in Detail

Issue Details on the Structure Board

Structure Board can now display full details of the selected issue in a right-hand panel, much like JIRA's issue navigator does. Just click the link to the issue in the Summary or Key column, or click the button in the top right corner to open issue details panel.

This feature lets you view full details of an issue and make any changes you'd usually make on the issue page, without leaving Structure Board and still observing the structure of all issues.

Documentation: Viewing Issue Details(see page 108)

Keyboard Shortcuts: **O** (toggle details, switch keyboard focus to the details panel), **Shift+O** (toggle details without switching focus)

Work Logged Column

The new "Work Logged" column displays the total work logged over a specified period of time (unlike "Time Spent" field, which shows the all-time total work logged).

The column supports aggregation, so you can see the total work logged over a specific period for all sub-issues.

Documentation: Work Logged Column(see page 315)

Issue Removal Notification

When issues are removed from the structure (either by pressing Delete key, or by clicking the Remove button in the Structure toolbar), a notification is shown briefly at the top of the page, with an Undo link. This should help with noticing and reverting the change if you hit Delete accidentally.

---

417 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Documentation: Removing Items from Structure(see page 107)

Archiving Old Structures

When you have a structure that is no longer updated and rarely looked at, but you still need to keep it for possible future reference, you can archive such structure. As a result, the structure is no longer suggested in the menus, it is made read-only and all synchronizers on the structure are deactivated.

Documentation: Archiving a Structure(see page 372)

Changing Structure Owner

JIRA administrators can now change the owner of a structure by using Configure link on the Manage Structure page.

The owner of a structure always has full Control access to the structure, so this feature helps when full access must be transferred from one person to another, such as when somebody leaves a project or the company.

Copying Structure with Synchronizers

When you make a copy of a structure (for example, when copying/cloning a template structure), you can select "Copy Synchronizers" option. As a result, the original structure's synchronizers are copied and installed on the new structure.

The copied synchronizers are initially disabled, so you need to manually Resync & Enable them.

Documentation: Copying Synchronizers(see page 538)

Other Notable Improvements

Structure 2.9 contains quite a few other improvements and fixes, including:

- Structure gadget can be maximized on a JIRA Dashboard[418]
- Allow specifying custom title of the Structure gadget[419]
- Export to Excel toolbar button is now enabled on Mac OS X[420]
- Compact view of the Issue Key column, showing project icon and issue number[421]

Our public JIRA contains the full list of issues resolved in Structure 2.9[422].

7.27.1.3 Supported JIRA Versions

Structure 2.9 supports JIRA versions 6.1—6.3+.

7.27.1.4 Changes for Developers

We've made some backwards-compatible additions to the Java API that are related to structure archiving and view settings for the Structure Board page with issue details enabled. For details, please see API Changes in Structure 2.9(see page 765).

---

418 https://jira.almworks.com/browse/HJ-1457
419 https://jira.almworks.com/browse/HJ-1240
420 https://jira.almworks.com/browse/HJ-1377
421 https://jira.almworks.com/browse/HJ-1041
422 https://jira.almworks.com/secure/StructureBoard.jspa?s=197

7.27.1.5 Installation

The plugin may be installed from the Atlassian Marketplace[423] or from the **Administration | Add-ons** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.1.6 Upgrading from a Previous Version

- If you're upgrading from version 2.4.3 or earlier, please read Structure 2.5 Release Notes(see page 777).
- If you're upgrading from version 2.2.1 or earlier, please read Structure 2.3 Release Notes(see page 792).

Upgrade procedure is simple:

1. Create backup of Structure data. Use **Administration | Structure | Structure Backup**. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

Downgrading to Structure 2.8 or Earlier Versions

Please note that View Settings (associated and default views, global and per structure) are migrated to a new format when you upgrade to Structure 2.9. If you downgrade to Structure 2.8, you will lose changes made to the view settings while working in Structure 2.9 or later.

7.27.1.7 API Changes in Structure 2.9

Minor Java API Release

With this release we add a few new methods to the Java API. The changes are backwards-compatible.

JIRA Version	New API Version
6.x	8.6.0

See Structure API Versions(see page 883) for full version information and downloads.

Compatible Changes in the Java API

Most of the API changes in this version support the new ability to archive structures(see page 372). There also a few other changes as well.

New methods in `Structure`

- `setArchived(boolean)`

423 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

- `isArchived()`

New constant in `StructureError`

- `ARCHIVED_STRUCTURE_EDIT_DENIED`

New method in `StructureFavoriteManager`

We've added an overload to method `getFavorites(User)`:

- `getFavorites(User, boolean includeArchivedStructures)`

Note that `getFavorites(User)` doesn't include archived structures in the result.

New methods in `StructureManager`

We've added overloads to some methods that return a list of structures by adding a flag to include archived structures in the result. Note that the original methods do not include archived structures in the result.

- `getAllStructures(User, PermissionLevel, boolean, boolean includeArchived)`
- `getStructuresByName(String, User, PermissionLevel, boolean, boolean includeArchived)`
- `hasStructuresForUser(User user, PermissionLevel, boolean includeArchived)`

Note that the existing method, `getRecentlyUpdatedStructures(User, PermissionLevel, int)`, doesn't include archived structures in the result.

We have also added a new method:

- `isArchived(Long)`

New constant in `StructurePage`

To support viewing issue details on the structure board(see page 108), we've added a new page type:

- `STRUCTURE_BOARD_WITH_DETAILS`

Changed constants in ViewSettings

Another change to support viewing issue details on the structure board(see page 108) — we've updated the following constants to include the new structure page type:

- `ALL_PAGES`
- `PAGES_WITH_DEFAULT_VIEW`

7.27.2 Structure 2.8 Release Notes

7 July 2014

Structure 2.8 adds support for JIRA 6.3 and introduces a Structure-based workflow validator, among other improvements and fixes.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[424]

7.27.2.1 Version Highlights

- Support for JIRA 6.3
- Workflow integration
- "Open Structure Board" Link
- Synchronization permission
- Other improvements and fixes

7.27.2.2 Changes in Detail

Support for JIRA 6.3

With this version we add support for the upcoming JIRA 6.3. We've fixed quite a few compatibility problems while testing Structure 2.8 against the latest JIRA 6.3 milestone, so if you're planning to upgrade your JIRA instance, please upgrade Structure as well.

We also drop support for JIRA 6.0. The last version to officially support JIRA 6.0 is Structure 2.7. Although there are no known issues that prevent Structure 2.8 from working on JIRA 6.0, we haven't tested this configuration.

Workflow Integration

Structure 2.8 introduces a Structure-based workflow validator, which can block a workflow transition if the issue being transitioned doesn't match an S-JQL(see page 266) query. For example, it can prevent an issue from being resolved if it has unresolved sub-issues in a structure.

Documentation: Workflow Integration(see page 494)

"Open Structure Board" Link

To quickly navigate to the Structure Board from any other page with a Structure widget, you can now use the "Open" link in the bottom right corner of the widget. It opens the currently viewed structure.

Documentation: JIRA Pages with Structure(see page 83)

Synchronization Permission

Structure administrators can now specify which user groups or project roles are allowed to configure synchronizers(see page 514). This new permission is required to:

- install, uninstall, enable, or disable a synchronizer;
- perform manual resync;
- import or export a structure.

By default, as in earlier versions, any user having **Control** access to a structure can use synchronization on that structure.

Documentation: Changing Permission to Manage Synchronizers(see page 522)

---

424 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Other Improvements and Fixes

This release also contains a number of smaller improvements and fixes, related primarily to the Links synchronizer, the Structure gadget, and S-JQL search.

For more information please refer to the full list of issues resolved in Structure version 2.8[425] in our public JIRA.

7.27.2.3 Supported JIRA Versions

Structure 2.8 supports JIRA versions 6.1—6.3.

7.27.2.4 Changes for Developers

We've made some backwards-compatible additions to the Java API in order to implement the workflow validator and the synchronization permission, so the minor API version has been updated. For details, see API Changes in Structure 2.8(see page 770).

7.27.2.5 Installation

The plugin may be installed from the Atlassian Marketplace[426] or from the **Administration | Add-ons** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.2.6 Upgrading from a Previous Version

- If you're upgrading from version 2.4.3 or earlier, please read Structure 2.5 Release Notes(see page 777).
- If you're upgrading from version 2.2.1 or earlier, please read Structure 2.3 Release Notes(see page 792).
- If you're using GreenHopper 6.1 and the new epics functionality, and upgrading from version 2.0 or earlier, please read Structure 2.1 Release Notes(see page 797).
- If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801).

Upgrade procedure is simple:

1. Create backup of Structure data. Use **Administration | Structure | Structure Backup**. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.27.2.7 Structure 2.8.1 Release Notes

17 July 2014 Structure 2.8.1 fixes rank synchronization with JIRA Agile 6.4.

425 https://jira.almworks.com/secure/StructureBoard.jspa?s=192
426 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[427]

Patch Release

This is a patch release based on version 2.8, which fixes a compatibility problem with JIRA Agile 6.4.

The following issues have been resolved:

- HJ-1823 JIRA Agile 6.4: Rank synchronization doesn't work[428]
- HJ-1826 Send "obsolete Rank field" notifications to Agile synchronizer owners[429]

Upgrade is recommended for all customers who have JIRA Agile 6.4+ installed and use the JIRA Agile synchronizer.

Rank Synchronization Fix

Rank synchronization stopped working after the ranking changes in JIRA Agile 6.4. This release restores the rank synchronization functionality.

Synchronizers based on Agile Boards should start working normally after upgrading to Structure 2.8.1. We recommend synchronizer owners to perform a manual resync after the upgrade to take care of any rank changes that may have been missed.

Custom Agile synchronizers will require reconfiguration, and their owners will receive e-mail notifications from Structure. The recommended course of action is to edit the synchronizer configuration (select the new **Rank** field) then resync and enable the synchronizer.

Supported JIRA Versions

Structure 2.8.1 supports JIRA versions 6.1 – 6.3.x.

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.8 Release Notes(see page 766) for details.

7.27.2.8 Structure 2.8.2 Release Notes

21 July 2014 Structure 2.8.2 fixes compatibility problems with JIRA 6.3.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[430]

Patch Release

Structure 2.8.0 added support for JIRA 6.3. Unfortunately, a number of critical compatibility problems that had been fixed in version 2.8.0, appeared again as regressions in version 2.8.1. This patch fixes the regressions and restores compatibility with JIRA 6.3.

427 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
428 https://jira.almworks.com/browse/HJ-1823
429 https://jira.almworks.com/browse/HJ-1826
430 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is recommended for all customers who use Structure with JIRA 6.3.

Supported JIRA Versions

Structure 2.8.2 supports JIRA versions 6.1 – 6.3.x.

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.8 Release Notes(see page 766) for details.

7.27.2.9 Structure 2.8.3 Release Notes

25 July 2014 Structure 2.8.3 fixes a problem with Agile boards in JIRA 6.3 and JIRA Agile 6.4.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[431]

Patch Release

This is a patch release based on version 2.8.2, which fixes a critical compatibility problem:

- HJ-1849 Workflow and other issue actions don't work on Agile boards[432]

The problem is present in Structure versions 2.8 and 2.8.2 when used together with JIRA 6.3 or later and JIRA Agile 6.4 or later.

Upgrade is recommended for all customers who use Structure with JIRA 6.3 and JIRA Agile 6.4.

Supported JIRA Versions

Structure 2.8.3 supports JIRA versions 6.1 – 6.3.x.

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.8 Release Notes(see page 766) for details.

7.27.2.10 API Changes in Structure 2.8

Minor Java API Release

With this release we add a few new methods to the Java API. The changes are backwards-compatible.

431 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

432 https://jira.almworks.com/browse/HJ-1849

JIRA Version	New API Version
6.x	8.5.0

See Structure API Versions(see page 883) for full version information and downloads.

Compatible Changes in the Java API

New methods in `StructureConfiguration`

New methods have been added to the `StructureConfiguration` class to support the new synchronization permission(see page 522):

- `isSynchronizationEnabledForAnyone()`
- `getSynchronizationPermissionSubjects()`
- `setSynchronizationEnabledForAnyone(boolean)`
- `setSynchronizationPermissionSubjectsEncoded(String)`
- `isSynchronizationAllowed(User)`

New methods in `StructureQuery`

New methods have been added to the `StructureQuery` class to support workflow integration(see page 494):

- `getSanitizedQueryString(User)`
- `checkIssue(Long, Forest, User, boolean)`
- `checkIssueIndex(int, Forest, User, boolean)`

7.27.3 Structure 2.7 Release Notes

26 February 2014

Structure 2.7 adds support for JIRA 6.2, integrates with the standard "Create Issue" dialog, and allows you to reconfigure installed synchronizers.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[433]

7.27.3.1 Version Highlights

- Support for JIRA 6.2
- Create Issue dialog integration
- Reconfigurable synchronizers
- Other improvements and fixes

433 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.27.3.2 Changes in Detail

Support for JIRA 6.2

Structure 2.7 is the first version to support JIRA 6.2. Previous versions are known to have critical compatibility problems with JIRA 6.2, so if you upgrade your JIRA, please upgrade Structure as well.

JIRA 5.2 series is no longer supported. The last version to support JIRA 5.2 is Structure 2.6.1(see page 776), which contains all of the bug fixes from Structure 2.7.

Create Issue Dialog Integration

It is now possible to create new issues in your structures using JIRA's standard "Create Issue" dialog. Furthermore, Structure lets you switch between inline editors and the dialog during issue creation, so you can set all the field values you want without adding too many columns to the structure grid.

Finally, when you use the standard "Create Issue" button on the Structure Board, the new issue will be added to your current structure by default.

Documentation: Creating New Issues(see page 110)

Reconfigurable Synchronizers

Structure 2.7 lets you change the parameters of installed synchronizers. There's a couple of things to keep in mind:

- If automatic synchronization is enabled for a synchronizer, Structure will disable it before editing parameters. When you're done, you will need to re-synchronize and enable the synchronizer explicitly.
- The user who changes synchronizer parameters becomes its new owner by default. Only JIRA administrators can change synchronizer parameters while keeping its current owner.

Documentation: Modifying Synchronizer(see page 518)

Other Improvements and Fixes

This release also contains a number of smaller improvements and fixes, related primarily to synchronization, the Progress column, and administration. We also introduce alternative gadgets(see page 771) that support Internet Explorer 8 and 9 in JIRA 6.0—6.1.

For more information please refer to the full list of issues resolved in Structure version 2.7[434] in our public JIRA.

7.27.3.3 Supported JIRA Versions

Structure 2.7 supports JIRA versions 6.0—6.2. If you need a version for JIRA 5.2, we recommend that you upgrade to Structure 2.6.1.

7.27.3.4 Changes for Developers

We've made some backwards-compatible additions to the Java API in order to support synchronizer reconfiguration, so the minor API version has been updated. For details, see API Changes in Structure 2.7(see page 773).

---

434 https://jira.almworks.com/secure/StructureBoard.jspa?s=176

7.27.3.5 Installation

The plugin may be installed from the Atlassian Marketplace[435] or from the **Administration | Add-ons** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.3.6 Upgrading from a Previous Version

- If you're upgrading from version 2.4.3 or earlier, please read Structure 2.5 Release Notes(see page 777).
- If you're upgrading from version 2.2.1 or earlier, please read Structure 2.3 Release Notes(see page 792).
- If you're using GreenHopper 6.1 and the new epics functionality, and upgrading from version 2.0 or earlier, please read Structure 2.1 Release Notes(see page 797).
- If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801).

Upgrade procedure is simple:

1. Create backup of Structure data. Use **Administration | Structure | Structure Backup**. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.27.3.7 API Changes in Structure 2.7

Minor Java API Release

With this release we're making a couple of backwards-compatible additions to the API in order to support editable synchronizers. We have also clarified the contracts of `com.almworks.jira.structure.api.sync` package by adding annotations to its interfaces and classes.

JIRA Version	New API Version
6.x	8.4.0

See Structure API Versions(see page 883) for full version information and downloads.

Compatible Changes in the Java API

- The new `StructureEditableSynchronizer` interface extends `StructureSynchronizer`. Its single method, `addFormParameters()`, is basically the inverse of `StructureSynchronizer.buildParametersFromForm()`. Please implement it properly in your custom synchronizer class in order to make it editable.
- The new `StructureSyncManager.updateSynchronizer()` method lets you programmatically change the owner and parameters of an installed synchronizer.

---

435 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

- `@PublicApi` and `@PublicSpi` annotations have been added to the interfaces and classes in `com.almworks.jira.structure.api.sync` to clarify compatibility contracts.
- `@NotNull` and `@Nullable` annotations have been added to the unannotated methods and parameters in `com.almworks.jira.structure.api.sync` to clarify nullability contracts.

7.27.4 Structure 2.6 Release Notes

24 January 2014

Structure 2.6 introduces a new collaboration feature – shared perspectives – and contains a number of improvements and fixes related primarily to synchronization, progress calculation, and administration.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[436]

7.27.4.1 Version Highlights

- Shared perspectives
- Better protection from synchronizer conflicts
- Global Structure permissions for project roles
- Synchronization with specific Agile Boards
- Progress column improvements and fixes

7.27.4.2 Changes in Detail

Sharing Perspectives

Did you ever want to show someone a part of a large structure, with all the columns and other presentation aspects configured exactly as you see them? Now you can share a Structure perspective. A **perspective** is a short link encompassing everything you see on the Structure Board – the structure, its visible part, all the columns, expanded, collapsed and selected issues, search terms, and more. You can create a perspective with a single click and send the link to other people or save it as a bookmark to return to it later.

Documentation: Sharing a Perspective(see page 92)

Protection from Synchronizer Conflicts

It is possible to accidentally configure a pair of synchronizers that would revert one another's changes endlessly, leading to degraded performance and a lot of noise in the Activity Streams. Protection against such conflicts has been around since version 2.0, but with this release we make it more prominent; by default, Structure will disable the conflicting synchronizers and send e-mail notifications to the users who installed them.

Documentation: Protection from Synchronizer Cycles(see page 522)

Permissions for Project Roles

JIRA administrators can now grant access to Structure and the permission to create new structures to particular project roles – in addition to user groups. This can greatly simplify Structure permission configuration in certain

436 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

scenarios. Please see Restricting User Access to Structure(see page 478) and Changing Permission to Create New Structures(see page 479) for more information.

Synchronization with Agile Boards

It is now possible to synchronize with a specific Agile Board – JIRA Agile synchronizer(see page 774) will use its query and ranking. We have also fixed a few bugs in Agile and Links synchronizers.

Progress Column Improvements

The configurable Progress column was introduced in Structure 2.5. In this release we've made several improvements to the progress aggregation algorithm to address the cases, noticed by our customers, where it yielded unpredictable or confusing results. We have also fixed a bug in the Icons column.

Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 2.6[437].

7.27.4.3 Supported JIRA Versions

Structure 2.6 supports JIRA versions 5.2 — 6.1. There are two separate downloads: one for JIRA 5.2, another for JIRA 6.x.

7.27.4.4 Changes for Developers

The API changes in this version are minimal, but the minor API versions have been updated. For details, see API Changes in Structure 2.6(see page 777).

7.27.4.5 Installation

The plugin may be installed from the Atlassian Marketplace[438] or from the **Administration | Add-ons** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.4.6 Upgrading from a Previous Version

- If you're upgrading from version 2.4.3 or earlier, please read Structure 2.5 Release Notes(see page 777).
- If you're upgrading from version 2.2.1 or earlier, please read Structure 2.3 Release Notes(see page 792).
- If you're using GreenHopper 6.1 and the new epics functionality, and upgrading from version 2.0 or earlier, please read Structure 2.1 Release Notes(see page 797).
- If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801).

437 https://jira.almworks.com/secure/StructureBoard.jspa?s=173
438 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade procedure is simple:

1. Create backup of Structure data. Use **Administration | Structure | Structure Backup**. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.27.4.7 Structure 2.6.1 Release Notes

25 February 2014

Structure 2.6.1 for JIRA 5.2 fixes two bugs in the search feature, along with a few other fixes and improvements.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[439]

Patch Release

This is a patch release based on version 2.6. The most prominent fixes are related to the search functionality:

- HJ-1691 JavaScript exceptions in search.js module[440] – an error message in the browser if you open Structure's search panel and leave the page unattended for a long time.
- HJ-1692 Search result secondary panel disappears when search result changes[441]

Our public JIRA contains the full list of issues resolved in version 2.6.1[442].

Upgrade is recommended for all customers who have active maintenance subscription.

Supported JIRA Versions

Structure 2.6.1 supports JIRA 5.2.x only! Upgrade to Structure 2.7 on JIRA 6.

This is the last version supporting JIRA 5.2 series. Structure 2.7 supports JIRA 6.0–6.2 and contains all the fixes from version 2.6.1.

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.6 Release Notes(see page 774) for details.

439 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
440 https://jira.almworks.com/browse/HJ-1691
441 https://jira.almworks.com/browse/HJ-1692
442 https://jira.almworks.com/secure/StructureBoard.jspa?s=177

7.27.4.8 API Changes in Structure 2.6

Minor Java API Release

The changes in the API in this release are minor. We had to bump the minor version of the API because we've added two constants to the `StructureError` enum.

JIRA Version	New API Version
5.x	7.7.0
6.x	8.3.0

See Structure API Versions(see page 883) for full version information and downloads.

Compatible Changes in the Java API

- Two constants have been added to the `StructureError` enum. They represent perspective-related errors.
- `PermissionSubject.ProjectRole` class can accept 0 as project ID meaning "all projects" in the global Structure permissions context.

7.27.5 Structure 2.5 Release Notes

26 September 2013

Structure 2.5 adds support for JIRA 6.1 and has improved, highly customizable Progress column, calculation of totals for all numeric fields, and a few other improvements. It also introduces Java and JavaScript APIs for developing your own columns.

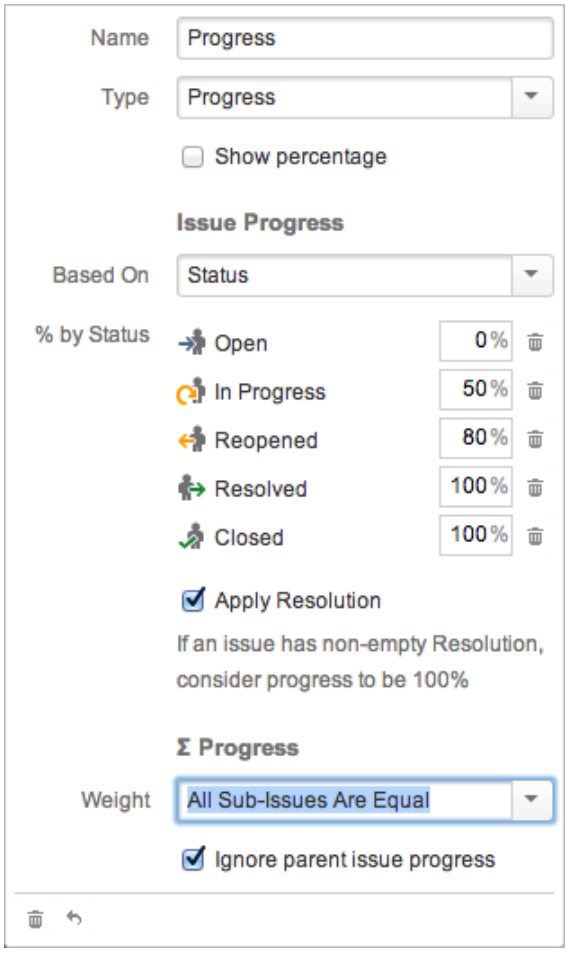
Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[443]

7.27.5.1 Version Highlights

- Progress column that supports story points, status-based calculation and custom percent field.
- Calculating totals for any numeric field.
- Structure API for adding new Structure column types from other plugins.

7.27.5.2 Changes in Detail

443 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure



Configurable Progress Column

The new Progress column supports multiple ways to calculate progress, such as:

- Aggregating sub-issues progresses based on their Story Points value;
- Determining progress of a single issue based on its Status and a customizable % value assigned to that Status;
- Using a custom field to specify completion % and aggregating that value up;
- Ignoring issues that have undefined completion % or a Status that isn't assigned % value;
- Using time tracking information for calculating progress (this was the only option available in the previous versions);
- Turning on/off 100% progress value for issues that have a non-empty Resolution.

To try the new functionality, click the arrow marker on the Progress column and play with possible configurations.

Documentation: Progress Column(see page 300)

Calculating Sum for Any Numeric Field

Any numeric field can now be aggregated up the structure, and the total value can be displayed for parent issues. Click the arrow marker on the numeric column and turn on **Sum over Sub-Issues**, or select the pre-configured totals column from the **Add Column** drop-down.

Documentation: Field Columns(see page 297)

Renaming Columns and New Column Configuration Panel

Each column type now may have multiple customizable options, including column name. The column names and other options are parts of the view specification, which can be saved and shared as a view(see page 341).

To view and change column options, click the little arrow marker in the Structure widget header – it appears if you move mouse pointer over the column header.

To accommodate the new configuration panel, we had to make certain changes to the already existing functionality:

- We have moved the remove column button (**x** icon) to the bottom of the column configuration panel.
- Instead of changing one column into another column via the drop-down (which used to be in the place of the column configuration panel), you can now change the column type and set all the options to achieve the same result.
- We have removed the **reorder** marker from the column header to conserve horizontal space. This is only a cosmetic change – you can still reorder columns using drag and drop, of course.

Documentation: Customizing Columns(see page 331)

New Column Extension API

Using the newest Structure API, it's now possible to create your own plugin that would extend Structure by adding new column types.

As an example, we have created a Status Bar Column Plugin(see page 965), a plugin that provides a column that shows a colored bar based on the distribution of statuses in sub-issues.

Documentation: Structure Developer's Guide(see page 842)

Direct S-JQL Search in the Structure Widget

It's now possible to use Structured JQL(see page 266) directly in the Structure's search bar. (Previously, you had to wrap it in the `structure()` JQL function.)

The search will apply to the currently visible structure.

Documentation: Search(see page 180)

Turn Off Description in the Summary field

You can now turn off showing Description in the main Summary field – use the arrow marker for the column options.

Documentation: Summary Column(see page 296)

Editing Value in the Aggregate Column

When a field column displays a total value for an issue (sum of a numeric or a time-tracking field over sub-issues), it's now possible to edit that issue's own value of that field in the usual way – by double-clicking or using keyboard to start editing a cell(see page 114).

This is a small convenience, which can save you from displaying two columns instead of one.

Other Improvements and Notable Bugs Fixed

- HJ-1388 Special Page for S-JQL Troubleshooting[444]
- HJ-1471 Links Synchronizer: possibility to switch to slower method of creating/deleting links, but which leaves records in the issue history[445]

7.27.5.3 Supported JIRA Versions

Structure 2.5 supports JIRA versions 5.2 — 6.1. There are two separate downloads: one for JIRA 5.x, another for JIRA 6.x.

JIRA versions 5.0 and 5.1 are no longer officially supported. While the Structure build for JIRA 5.x will likely work on JIRA 5.0 and JIRA 5.1, we didn't do proper testing and recommend to either upgrade JIRA, or use a previous version of Structure.

7.27.5.4 Changes for Developers

The new version of Structure API lets you create your own columns, but also can be used to calculate arbitrary aggregate values over a structure.

For details, see API Changes in Structure 2.5(see page 784).

7.27.5.5 Installation

The plugin may be installed from the Atlassian Marketplace[446] or from the **Administration | Add-ons** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.5.6 Upgrading from a Previous Version

- If you're upgrading from version 2.2.1 or earlier, please read Structure 2.3 Release Notes(see page 792).
- If you're using GreenHopper 6.1 and the new epics functionality, and upgrading from version 2.0 or earlier, please read Structure 2.1 Release Notes(see page 797).
- If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801).

View Specification Upgrade

In Structure 2.5, we have changed how certain columns work. The **TP** column (type, priority) has become Icons Column(see page 299), and Progress Column(see page 300) has been largely remade.

We have tried to make the upgrade experience as smooth as possible. The old columns should be converted to the new columns on the fly as Structure widget is opened. When you save changes to a view, the new view specification will refer to the new columns.

---

444 https://jira.almworks.com/browse/HJ-1388
445 https://jira.almworks.com/browse/HJ-1471
446 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

If you downgrade after saving any views, the old version of Structure might not show some of the columns.

Upgrade Procedure

Upgrade procedure is simple:

1. Create backup of Structure data. Use **Administration | Structure | Structure Backup**. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.

   Note for early adopters: if you have a pre-release version of Structure 2.5 installed, you need to uninstall it first, and only then install the released version. All structure data will be kept.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.27.5.7 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 2.5[447].

7.27.5.8 Structure 2.5.1 Release Notes

31 October 2013

Structure 2.5.1 fixes an issue with work logging and several issues with keyboard shortcuts and synchronization.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[448]

Patch Release

This is a patch release based on version 2.5. It contains a number of bug fixes and improvements related to work logging, keyboard shortcuts, and synchronization with JIRA Agile.

Upgrade is recommended for all customers who have active maintenance subscription.

Work Logging Fix

The following critical issue has been fixed:

- HJ-1542 Work log may be placed under a wrong issue when entered with a button on a time tracking section of the issue page[449] — this only affects JIRA 6 or newer.

Keyboard Shortcut Fixes

The following keyboard-related bugs have been fixed:

447 https://jira.almworks.com/secure/StructureBoard.jspa?s=167
448 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
449 https://jira.almworks.com/browse/HJ-1542

- HJ-1505 Comment shortcut doesn't work[450]
- HJ-1507 View Full Size Image shortcut doesn't work[451]
- HJ-1523 Several keyboard shortcuts don't work in the Gadget[452]

Synchronization Fixes and Improvements

Links and Agile synchronizers now react to changes made via Edit Issue and workflow dialogs. Other Agile synchronizer improvements include:

- HJ-1295 In Agile synchronizer, don't link sub-tasks to epics if "Force Sub-tasks" is enabled[453]
- HJ-1072 Removed Epic may be restored when sub-issue is moved in structure within the same parent[454]
- HJ-1478 Epic Sync: Child issue isn't moved on the structure after removing epic[455]

Other Improvements

Our public JIRA contains the full list of issues resolved in version 2.5.1[456]. One of them is notable:

- HJ-1525 Bulk change should process the issues in structure order[457] — this is important if you use Xporter for JIRA.

Supported JIRA Versions

Structure 2.5.1 supports JIRA versions 5.2 – 6.1.x. There are two separate downloads: one for JIRA 5, another for JIRA 6.

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.5 Release Notes(see page 777) for details.

7.27.5.9 Structure 2.5.2 Release Notes

1 November 2013

Structure 2.5.2 fixes a critical issue with JIRA Agile synchronization.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[458]

Patch Release

This is a patch release based on version 2.5.1. It fixes one critical and one minor issue with JIRA Agile synchronization:

---

450 https://jira.almworks.com/browse/HJ-1505
451 https://jira.almworks.com/browse/HJ-1507
452 https://jira.almworks.com/browse/HJ-1523
453 https://jira.almworks.com/browse/HJ-1295
454 https://jira.almworks.com/browse/HJ-1072
455 https://jira.almworks.com/browse/HJ-1478
456 https://jira.almworks.com/secure/StructureBoard.jspa?s=169
457 https://jira.almworks.com/browse/HJ-1525
458 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

- HJ-1548 JIRA Agile synchronizer pulling issues onto structure board incorrectly using epic link[459]
- HJ-1550 Editing Epic Link from the issue page doesn't trigger JIRA Agile synchronizer[460]

Upgrade is recommended for all Structure 2.5.1 users who have JIRA Agile synchronizers enabled.

Supported JIRA Versions

Structure 2.5.1 supports JIRA versions 5.2 – 6.1.x. There are two separate downloads: one for JIRA 5, another for JIRA 6.

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.5 Release Notes(see page 777) for details.

7.27.5.10 Structure 2.5.3 Release Notes

22 November 2013

Structure 2.5.3 fixes a few issues with synchronization and improves performance on very large structures.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[461]

Patch Release

This is a patch release based on version 2.5.2. It contains a number of performance improvements and bug fixes related to synchronization.

Upgrade is recommended for all customers who have active maintenance subscription.

Synchronization Fixes and Improvements

The following synchronization-related issues have been fixed:

- HJ-1580 JIRA Agile synchronizer incorrectly moves issues without epics to the top[462]
- HJ-1581 JIRA Agile synchronizer should "bubble up" the issues that lose epics instead of moving them to the top level[463]
- HJ-1573 DefaultLinkProcessor does not report link creation/removal events[464]

Aggregate Performance Improvements

In previous versions, overall JIRA performance could degrade in the presence of large structures (containing 20,000 issues and more), due to massive access checks and aggregate value calculations. Both performance and responsiveness have been improved in this version. Please see the following issues in our JIRA:

459 https://jira.almworks.com/browse/HJ-1548
460 https://jira.almworks.com/browse/HJ-1550
461 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
462 https://jira.almworks.com/browse/HJ-1580
463 https://jira.almworks.com/browse/HJ-1581
464 https://jira.almworks.com/browse/HJ-1573

- HJ-1539 Performance degradation due to excessive load via Structure aggregates[465]
- HJ-1538 NPE in AggregateCache[466]

Supported JIRA Versions

Structure 2.5.3 supports JIRA versions 5.2 – 6.1.x. There are two separate downloads: one for JIRA 5, another for JIRA 6.

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.5 Release Notes(see page 777) for details.

7.27.5.11 API Changes in Structure 2.5

Javascript API Release

In this release we're adding JavaScript API to customize client-side functionality of Structure. As of Structure 2.5, it allows to create custom column types for the Structure widget. Please refer to Structure JavaScript API Reference(see page 944) for more information.

Minor Java API Release

Structure API has been expanded — we've added support for custom columns for the Structure widget. This part of the API should supply the necessary data to the JavaScript API.

JIRA Version	New API Version
5.x	7.6.0
6.x	8.2.0

See Structure API Versions(see page 883) for full version information and downloads.

Compatible Changes in the Java API

New package `com.almworks.jira.structure.api.column`

This package allows to define components that support custom columns. You can extend data fields that are provided to the Structure widget (`data` subpackage), and how the values are exported to Excel and printable format (`export`, `excel`, and `printable` subpackages.)

465 https://jira.almworks.com/browse/HJ-1539
466 https://jira.almworks.com/browse/HJ-1538

New package `com.almworks.jira.structure.api.aggregate`

This package allows to implement a custom column that aggregates values over structure through the means of implementing the `Aggregate` interface. There's a caching layer, so values will be recalculated only for the changed issues and all dependent issues.

Notably, progress aggregates such as the Progress Column(see page 300) are now implemented through this package (see `progress` subpackage.)

Deprecated classes

`ProgressCalculator` and `ProgressInfo` classes are now deprecated; their counterparts from the new `aggregate` package – `AggregateCalculator` and `progress.ProgressAggregateFactory` – should be used instead. The latter supports the new, highly customizable Progress Column(see page 300).

Other changes

- We've added methods for the new kinds of columns to `ViewSpecification`.
- Due to the ability to rename projects in JIRA 6, we've added project ID-based methods to `StructureConfiguration`.
- We've added new functions to extract data from JIRA issue to `JiraFunc`. They make for easy implementation of the new kinds of columns for a specific JIRA issue field.

New Plugin Module Types

To extend Structure with new columns, you need to declare the extending components in your `atlassian-plugin.xml` using the new module types – `structure-widget-extension`, `structure-issue-data-provider` and `structure-export-renderer-provider`. See Structure Plugin Module Types(see page 888) for reference or the Status Bar Column Plugin(see page 965) for an example.

7.27.6 Structure 2.4 Release Notes

14 June 2013

Structure 2.4 introduces S-JQL language for querying structures.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[467]

7.27.6.1 Structured JQL

In this version, we're happy to introduce a long-awaited feature — **Structured JQL** (S-JQL) language for finding issues in structures that satisfy some structure-based conditions. Here are just a few examples:

467 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Example	S-JQL Expression
Select "leaf" issues (those that don't have sub-issues)	`leaf`
Get top-level issues and 2nd-level issues only	`root or child of root`
Retrieve all Epics in a certain status and all of their sub-issues	`issue or ancestor in [type = Epic and status = Open]`
Find a closed issue that has an open sub-issue at any level	`[status = Closed] and descendant in [status = Open]`

To use S-JQL expression in a normal JQL (for example, in the Issue Navigator), you need to use `structure()` JQL function, for example:
`issue in structure("Enterprise Portfolio", "issue or ancestor in [type = Epic and status = Open]")`

There are a lot more examples in the S-JQL Cookbook(see page 267).

Potential applications of the S-JQL queries:

- Configuring GreenHopper's Agile board based on a structure and specific issues from it.
- Using in JQL filters for selecting specific parts of work breakdown.
- Creating workflow conditions based on the position of the issue in structure.
- Using in synchronizers for limiting scope.

Documentation: Structured JQL(see page 266)

7.27.6.2 Other Changes

Notable Fixes

- Fixed: HJ-1184 Status rollup synchronizer does not correctly respond to Delete Issue action[468]
- Fixed: HJ-1202 Send to Top / Send to Bottom don't work in Firefox[469]
- Fixed: HJ-1338 Multiple problems with IE8-IE9 and JIRA 6[470]

7.27.6.3 Supported JIRA Versions

Structure 2.4 supports JIRA versions 5.0.1 — 6.0+. There are two separate downloads: one for JIRA 5.x, another for JIRA 6.x.

468 https://jira.almworks.com/browse/HJ-1184
469 https://jira.almworks.com/browse/HJ-1202
470 https://jira.almworks.com/browse/HJ-1338

JIRA 5.0 (**not** 5.0.1 or later) is **not** supported anymore. Structure 2.3.0.jira5 is the last Structure version that supports JIRA 5.0.

7.27.6.4 Installation

The plugin may be installed from the Atlassian Marketplace[471] or from the **Administration | Add-ons** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.6.5 Upgrading from a Previous Version

- If you're upgrading from version 2.2.1 or earlier, please read Structure 2.3 Release Notes(see page 792).
- If you're using GreenHopper 6.1 and the new epics functionality, and upgrading from version 2.0 or earlier, please read Structure 2.1 Release Notes(see page 797).
- If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801).

Upgrade procedure is simple:

1. Create backup of Structure data. Use **Administration | Structure | Structure Backup**. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.

   Note for early adopters: if you have a pre-release version of Structure 2.3 installed, you need to uninstall it first, and only then install the released version. All structure data will be kept.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.27.6.6 Structure 2.4.1 Release Notes

2 August 2013

Structure 2.4.1 contains a fix for a critical bug and other minor fixes and improvements, and adds Structure section to the Issue Navigator detail view.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[472]

Patch Release

This is a patch release based on version 2.4. It fixes a rare but critical bug that could significantly slow down JIRA. Additionally, Structure section is now available on the detail view in the new Issue Navigator in JIRA 6. There are many other bug fixes and smaller improvements, most of which target user experience with Structure on the Issue Page.

471 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
472 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Upgrade is recommended for all customers who have a license with active maintenance and a compatible JIRA.

Fixed: some S-JQL queries run too slow and can cause OutOfMemoryError on the server

In Structure 2.4, we have introduced S-JQL(see page 266), a way to find issues that satisfy structure-based conditions. Since then, we have discovered and fixed a major performance problem. The implementation of some relations(see page 0) (`prevSibling`, `nextSibling`, and `sibling`) was inefficient, so that running a `structure()` JQL query with these relations on a large structure (thousands of issues) could result in major performance degradation, in come cases causing OutOfMemoryErrors. The fix added in this version significantly improves the speed and reduces memory usage of these relations. We have improved other relations as well, so that almost all S-JQL queries now run faster.

Additionally, this version contains a fix for another problem[473] that could cause memory shortage over time.

Fixed: issues with keys starting on "I" display Structure widget even if the project is disabled for Structure and other problems on the issue page

This bug was introduced in Structure 2.4: in JIRA 6, if there is a project with key that starts with "I" or "i", Structure section is displayed on pages of issues in this project. We have fixed this along with many other problems with user experience on the issue page.

Notably, we have been receiving feedback regarding the adjusted time tracking section(see page 0), which indicates a confusion between "Include sub-issues", the option to sum tracked time over sub-issues in the selected structure, and "Include sub-tasks", the option added by JIRA to sum tracked time over JIRA sub-tasks. The option added by Structure now reads "Include structure sub-issues", reducing user confusion.

Structure section is now available in the Issue Navigator

Starting with JIRA 6.0, JIRA's Issue Navigator features the Detail View[474], so that users can work with issues in the context of the current filter. In this release, we add Structure section to the detail view.

Other changes

See these and other issues fixed in this release in the list below.

Error rendering macro 'gadget'

Error rendering gadget [https://jira.almworks.com/rest/gadgets/1.0/g/com.almworks.jira.structure:structure-gadget/gadgets/structure-gadget.xml]

Supported JIRA Versions

Structure 2.4.1 supports JIRA versions 5.0.1 – 6.0.5 and later. There are two separate downloads: one for JIRA 5.x, another for JIRA 6.x.

JIRA 5.0 (**not** 5.0.1 or later) is **not** supported anymore. Structure 2.3.0.jira5 is the last Structure version that supports JIRA 5.0.

473 https://jira.almworks.com/browse/HJ-1393
474 https://confluence.atlassian.com/display/JIRA060/Using+the+Issue+Navigator

Installation

The plugin may be installed from the Atlassian Marketplace[475] or from the **Administration | Add-ons** menu. Further information is available in the Administrator's Guide(see page 458).

Upgrading from a Previous Version

- If you're upgrading from version 2.2.1 or earlier, please read Structure 2.3 Release Notes(see page 792).
- If you're using GreenHopper 6.1 and the new epics functionality, and upgrading from version 2.0 or earlier, please read Structure 2.1 Release Notes(see page 797).
- If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801).

Upgrade procedure is simple:

1. Create backup of Structure data. Use **Administration | Structure | Structure Backup**. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.27.6.7 Structure 2.4.2 Release Notes

7 August 2013

Structure 2.4.2 contains a fix for the time tracking section on the issue page.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[476]

Patch Release

This is a patch release based on version 2.4.1. It fixes a problem that caused Time Tracking section on the issue page to disappear, even though the issue contained time tracking information.

The bug affected both JIRA 5 and JIRA 6 versions of the product.

Upgrade is recommended for all customers who have version 2.4.1 installed.

Supported JIRA Versions

Structure 2.4.2 supports JIRA versions 5.0.1 – 6.0.x. There are two separate downloads: one for JIRA 5, another for JIRA 6.

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.4 Release Notes(see page 785) for details.

475 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
476 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.27.6.8 Structure 2.4.3 Release Notes

1 September 2013

Structure 2.4.3 adds workarounds for grayed out Sprint panel and occasional Chrome browser crashes.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[477]

Patch Release

This is a patch release based on version 2.4.2. It provides the following fixes and workarounds:

- HJ-1448 Greenhopper 6.3 sprints grayed out[478] — we added a workaround for a problem[479] introduced in JIRA Agile (GreenHopper) 6.3, making it incompatible with Structure and a few other plugins.
- HJ-1434 Chrome bug workaround[480] — reducing the possibility of Google Chrome crashing when you use Structure widget.
- HJ-1446 Possible problems with rendering editors for Issue Type[481]

Upgrade is recommended for all customers who have active maintenance subscription.

On Crashing Google Chrome

Some of our users that work primarily in Google Chrome browser have been frequently annoyed by the browser crashing (the **Aw, Snap!** screen). Although it happened only when Structure widget was used, the problem had nothing to do with Structure Plugin's code — a crash is a system failure in Google Chrome, much like a blue screen on Windows. It just happened that Structure's code was causing this Chrome's bug to surface.

We have been able to add some workarounds in the Structure plugin that reduce *probability* of Chrome's crashing in a specific part of our code. The benefit is that Chrome may crash *less frequently* when you upgrade to version 2.4.3.

At the same time, we had productive communications with Chromium project team, and Chrome Canary (version 31.0 or later) also includes a real fix for at least one of the problems that made Chrome crash. If you can upgrade Chrome to that version, it should be more stable.

Supported JIRA Versions

Structure 2.4.3 supports JIRA versions 5.0.1 – 6.0.x. There are two separate downloads: one for JIRA 5, another for JIRA 6.

We're about to stop supporting JIRA 5.0 and JIRA 5.1 as we move forward to support the upcoming JIRA 6.1. If you are using one of those versions, please consider upgrading Atlassian JIRA when possible.

---

477 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
478 https://jira.almworks.com/browse/HJ-1448
479 https://jira.atlassian.com/browse/GHS-9730
480 https://jira.almworks.com/browse/HJ-1434
481 https://jira.almworks.com/browse/HJ-1446

Installation and Upgrade

Installation and upgrade are simple and done in the same way as for most other plugins. Please see Structure 2.4 Release Notes(see page 785) for details.

7.27.6.9 API Changes in Structure 2.4

Minor API Release

Structure API has been expanded – we've added interfaces that give you the access to the newly added S-JQL(see page 266) parsing and execution engine.

JIRA Version	New API Version
5.x	7.5.0
6.x	8.1.0

See Structure API Versions(see page 883) for full version information and downloads.

Compatible Changes

`StructureQuery` Class

This class represents a parsed S-JQL expression and allows you to execute it against a specified `Forest`. The result may be a list of `Issue` objects, or a list of `long` issue IDs.

`StructureQueryParser` Interface

This interface lets you parse a S-JQL expression and get an instance of `StructureQuery`. An instance of parser may be injected into your component or retrieved from `StructureServices`.

`StructureQueryBuilderFactory` Interface

This interface lets you construct a `StructureQuery` using expressive sequence of calls in Java – similar to `JqlQueryBuilder`. An instance of builder may be injected into your component or retrieved from `StructureServices`.

7.27.7 Structure 2.3 Release Notes

18 May 2013

Structure 2.3 adds support for JIRA 6 and includes a number of fixes.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[482]

7.27.7.1 JIRA 6 Support

JIRA 6 is a major new upgrade from Atlassian, with overhaul of the user interface and many other improvements. Structure plugin has been modified to fit nicely into the new JIRA package.

Most of the updates for JIRA 6 were adapting Structure user interface to the new look, but we also migrated Structure to the new JIRA platform to support such features as the ability to change login name.

7.27.7.2 Other Changes

Fixes

- Fixed: HJ-1169 Make Structure panel show for issue of type "Test", added by Zephyr[483]
- Fixed: HJ-1232 Use the same link to Log Work action as on the standard Time Tracking section on the issue page (even if it is replaced by Tempo)[484]
- Fixed: HJ-1242 Status Rollup synchronizer fails to find a global transition[485]
- Fixed: HJ-1265 Link synchronizer may erroneously move an issue to the top[486]
- Fixed: HJ-1308 Printable page loses view configuration when view is locally modified[487]

Anonymous Usage Statistics

We have introduced optional reporting of the usage statistics, which should help us better understand how Structure is used and focus on the important improvements. The statistics reporting is turned off by default, and Structure plugin may once suggest to a JIRA administrator to turn it on.

Documentation: Anonymous Usage Statistics(see page 497)

7.27.7.3 Supported JIRA Versions

Structure 2.3 supports JIRA versions 5.0 — 6.0+. There are two separate downloads: one for JIRA 5.x, another for JIRA 6.x.

482 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
483 https://jira.almworks.com/browse/HJ-1169
484 https://jira.almworks.com/browse/HJ-1232
485 https://jira.almworks.com/browse/HJ-1242
486 https://jira.almworks.com/browse/HJ-1265
487 https://jira.almworks.com/browse/HJ-1308

7.27.7.4 Installation

The plugin may be installed from the Atlassian Marketplace[488] or from the **Administration | Add-ons** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.7.5 Upgrading from a Previous Version

If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801). If you're using GreenHopper 6.1 and the new epics functionality, and upgrading from version 2.0 or earlier, please read Structure 2.1 Release Notes(see page 797).

If you have LDAP user directory with login names containing uppercase characters, the upgrade will involve Structure database migration. Structure backup is highly recommended!

Upgrade procedure is simple:

1. Create backup of Structure data. Use **Administration | Structure | Structure Backup**. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.

   Note for early adopters: if you have a pre-release version of Structure 2.3 installed, you need to uninstall it first, and only then install the released version. All structure data will be kept.
3. Monitor `catalina.out` or `jira-application.log` for warnings or errors.

7.27.7.6 API Changes in Structure 2.3

Major API Release

Due to JIRA 6 changes[489], we have to bump the major version of the Structure API in the version of Structure for JIRA 6. Although no considerable changes have been introduced, strictly speaking, the new Structure API for JIRA 6 is backwards-incompatible.

If you're using Structure API, please check these change notes and see if you need to make changes to your code. Most likely you don't.

If you limit the OSGi package import version(see page 883) to the 7.x range, you'll need to either expand the range to include 8.x, or build separate plugins for JIRA 5 and JIRA 6.

488 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure
489 https://developer.atlassian.com/display/JIRADEV/Preparing+for+JIRA+6.0

JIRA Version	New API Version
5.x	7.4.0
6.x	8.0.0

See Structure API Versions(see page 883) for full version information and downloads.

Incompatible Changes in Structure for JIRA 6

`PermissionRule` and `PermissionSubject` semantics change

Due to migration in JIRA 6 from using user login names to using user keys (to support changing login names), all objects that represent permission rules are using user keys now, while preserving serialization format and class methods.

This means that if you happen to serialize `PermissionRule` and `PermissionSubject` or use their `toEncodedString` or `fromEncodedString`, you'll need to migrate your persisted data (convert login names to lowercase). Structure plugin does that on its own for the data stored in the database and when restoring from backup.

Compatible Changes

Additional Executor in `StructureJobManager`

`StructureJobManager` can now execute tasks in multiple independent queues (executors). See javadocs for the details.

`StructureUtil` Methods for Migration

`StructureUtil` has a few new facade methods to help with migrating users – see `migrateUserNameToUserKey()`, `getUserByName()`, `getUserByKey()`.

7.27.8 Structure 2.2 Release Notes

28 March 2013

Starting with version 2.2, Structure plugin is available for purchase on the Atlassian Marketplace.

Download the Latest Version(see page 978)
Structure on the Atlassian Marketplace[490]

7.27.8.1 Atlassian Marketplace Availability

Structure Plugin can now be purchased directly from Atlassian via Atlassian Marketplace.

490 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Starting from version 2.2, Structure supports Atlassian Marketplace licenses, obtained or purchased from Atlassian and managed along with your other Atlassian licenses on my.atlassian.com[491]. JIRA administrators can now purchase a license or get an evaluation right in the JIRA's Plugin Manager.

Structure will continue to support licenses issued by ALM Works. You can use either ALM Works or Atlassian licenses - both are fully functional and cost the same.

Documentation: Setting Up Structure License(see page 473)

> **Attention Existing Customers: don't worry!** After upgrade to version 2.2, Plugin Manager will show you that Structure plugin is **"Unlicensed"** or perhaps **"Requiring Action"**, even though you have a valid license from ALM Works installed.
>
> That is due to the fact that Plugin Manager is not aware of the licenses issued by ALM Works. To verify the true status of the Structure license, please check **Administration | Structure | License Details** page. If it shows you that the license is OK, you can safely ignore the status of the license in Plugin Manager.

It is possible to convert your existing license, issued by ALM Works, into an Atlassian license. If you'd like to do so, please contact sales[492].

7.27.8.2 Other Changes

Minor fixes and improvements

Besides Marketplace support, this version contains several bug fixes and small improvements.

- Fixed: HJ-1151 Issue cloner fails to copy Due Date on JIRA with different date formats for input and output[493]
- Fixed: HJ-1155 Issues not in filter are not dimmed on IE8[494]
- Fixed: HJ-1190 Cloning may fail if Summary gets too long after adding prefix / suffix[495]
- HJ-1163 Remove conflicting shortcuts "j" and "k" from issue page[496]

7.27.8.3 Supported JIRA Versions

Structure 2.2 supports JIRA versions 5.0.1 — 5.2+.

JIRA 5.0 is also supported, but only if you first upgrade its Atlassian Universal Plugin Manager plugin from the bundled version (1.5.x) to at least version 2.0.1 or the latest version available on the Atlassian Marketplace. JIRA versions 5.0.1 and later already have the required version of Plugin Manager.

7.27.8.4 Installation

The plugin may be installed from the Atlassian Marketplace[497] or from the **Administration | Plugins** menu. Further information is available in the Administrator's Guide(see page 458).

---

491 https://my.atlassian.com
492 mailto:sales@almworks.com
493 https://jira.almworks.com/browse/HJ-1151
494 https://jira.almworks.com/browse/HJ-1155
495 https://jira.almworks.com/browse/HJ-1190
496 https://jira.almworks.com/browse/HJ-1163
497 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

7.27.8.5 Upgrading from a Previous Version

> ⚠ If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801).

> ⚠ If you're using GreenHopper 6.1 and the new epics functionality, and upgrading from version 2.0 or earlier, please read Structure 2.1 Release Notes(see page 797).

Migration from version 2.0—2.1.1 to version 2.2 is pretty straightforward. Structure Backup is recommended as a safety measure.

Proper upgrade sequence:

1. Create backup of Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the whole `structure/` sub-directory under JIRA home while Structure is disabled. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.

   > ⚠ Note for early adopters: if you have a pre-release version of Structure 2.2 installed, you need to uninstall it first, and only then install the released version. All structure data will be kept.

3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.

7.27.8.6 Structure 2.2.1 Release Notes

> ⓘ **12 April 2013**
>
> Structure 2.2.1 contains an important bug fix and minor improvements.

Download the Latest Version(see page 978)
Structure on Atlassian Marketplace[498]

Patch Release

This is a patch release based on version 2.2. It contains a fix for a potentially serious issue and several other improvements and fixes.

Upgrade is recommended for all customers who have a license with active maintenance and a compatible JIRA.

Fixed: Possible Loss of Structure Due To Broken JIRA Index

In Structure 2.1, we have introduced automatic daily maintenance(see page 490), which, among other things, sanitizes structures by removing references to issues that are deleted from JIRA. It turned out that in case JIRA has broken indexes (and requires re-indexing), this procedure can incorrectly identify some issues as deleted and remove them from structures.

The fix added in this version double-checks the existence of issues in an index-independent way.

---

498 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Although this bug[499] is critical, it is quite rare. You might have been affected by this issue if:

- You have Structure automatic daily maintenance enabled, including "optimizing structure" option. (It is enabled by default.)
- Your JIRA was left running with incorrect indexes during a long time or overnight. (By default, daily maintenance runs at 3 am.)
- The indexes were broken in a way that resulted in JQL searches like `id in (10000,10001,10002)` to return incorrect results. (A typical plugin installation/uninstallation does not break the index in that way.)

If you think you might have been affected, please check the consistency of your structures. You can use structure history(see page 375) or structure migration(see page 488) to partially restore the structures, if they are affected.

Administrator's User Interface Improvement for Current Customers

If you have a Structure license from ALM Works and you have upgraded to version 2.2, you may have noticed that the Plugin Manager started showing Structure as an unlicensed plugin.

This happens due to the fact that Structure can now be purchased from Atlassian, as well as from ALM Works – but Plugin Manager knows only about Atlassian licenses. You may have a perfectly valid license, issued by ALM Works, but still have Plugin Manager warn you about missing license (without otherwise affecting JIRA and Structure functionality).

To avoid confusion, for those who have ALM Works license installed, Structure now displays additional note on the Plugin Manager page, explaining this situation.

Other Fixes

- Fixed: HJ-1233 Progress may be calculated incorrectly for an issue that has resolved sub-issue, which has non-resolved sub-sub-issue[500]
- Fixed: HJ-1206 Progress for parent issues is not updated after editing time in a child issue[501]
- Fixed: HJ-1198 Quote escaping in JQL is lost when going from Issue Navigator to Structure Board[502]
- Fixed: HJ-1205 Issue type drop-down does not contain default type[503]
- Fixed: HJ-1223 Structure gets package-refreshed for no reason when installed or when JIRA is started[504]

Supported JIRA Versions

This version supports JIRA 5.0.1--5.2.10 and later.

7.27.9 Structure 2.1 Release Notes

15 February 2013

Structure 2.1 adds undo, automatic daily backups, support for GreenHopper 6.1, and contains quite a few minor improvements and bug fixes.

499 https://jira.almworks.com/browse/HJ-1211
500 https://jira.almworks.com/browse/HJ-1233
501 https://jira.almworks.com/browse/HJ-1206
502 https://jira.almworks.com/browse/HJ-1198
503 https://jira.almworks.com/browse/HJ-1205
504 https://jira.almworks.com/browse/HJ-1223

Download the Latest Version(see page 978)
Structure on Plugin Exchange[505]

7.27.9.1 Version Highlights

- Support for GreenHopper 6.1 with the new Epics
- Undo
- Automatic daily backup and maintenance

7.27.9.2 Changes in Detail

Support for GreenHopper 6.1

In version 6.1, GreenHopper team had introduced a new approach to epics, backed by a different set of custom fields, notably **Epic Link** and **Epic Name**. The old **Epic/Theme** field was deprecated and is used now only on classic boards. Structure 2.0 and earlier versions are not fully compatible with this change.

With Structure 2.1, you can continue using GreenHopper and Structure together:

- GreenHopper synchronizer can now work with the new epics, synchronizing structure with epic-story assignment on the Scrum Board. When installing the synchronizer, select **Scrum Board Epics** as the "Epic Field". The synchronizer can still work with older Epic/Theme field as well.
- When a new epic is created in the Structure Widget, and the Epic Name is not set by the user (it's a required field for epics), Structure automatically uses the value you have entered as the issue's Summary.
- Structure section on the issue details panel in GreenHopper now enters "Pinned" mode by default, showing you only the parent issues and sub-issues of the selected issue. You can pin and unpin Structure widget as usual.

If you are upgrading Structure and already have GreenHopper synchronizer installed, it will continue working with the old Epic/Theme field. To switch to the new Scrum Board Epics integration, you need to use **Manage Structures** page to delete or disable the old synchronizer and create a new one(see page 797).

Documentation: JIRA Agile (GreenHopper) Synchronizer(see page 797), Structure on Agile Boards(see page 91)

Undo

Structure 2.1 adds limited Undo functionality, which lets you revert a single recent change you've made to a structure. Not all operations can be undone, but those most frequently used – like removing issues from structure or drag-and-drop – can be reverted.

Documentation: Undoing Changes(see page 107)

Automatic Daily Backup and Maintenance

As Structure data is stored in its own database(see page 498), it needs to be backed up separately from the main JIRA backup. That also means that the default automatic backup run by JIRA does not cover Structure data.

Starting from this version, Structure can automatically run daily maintenance procedure at the time specified by JIRA administrator. The maintenance procedure creates a backup of Structure data and performs several optimizations on the Structure database.

---

505 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

Documentation: Automatic Structure Maintenance(see page 490)

Structure Can Be Selected as "My JIRA Home"

You can now select Structure Board as your default page in JIRA (on JIRA 5.1 or later).

Click the profile menu in the top right corner and select **Structure** under **My JIRA Home** directory. After that, if you open JIRA in browser using its general address, Structure Board will be shown automatically, with the latest structure you've been working with.

Other Notable Improvements and Bugs Fixed

- HJ-1057 Exclude archived versions from version fields when creating new issues (with copying categories) and cloning issues[506]
- Fixed: HJ-1069 Repetitive call to the same workflow action through operations dialog fails[507]
- Fixed: HJ-888 minus-minus shortcut does not work in Firefox 15[508]
- Fixed: HJ-1059 Migration failure in case exported XML contains deleted issue ids[509]
- Fixed: HJ-1060 Export to Excel / Printable page do not work if the selected view is "Issue Navigator"[510]
- Fixed: HJ-1093 Structure grid displays unsupported JIRA columns when "Columns" mode is turned on[511]
- Fixed: HJ-1100 Rank to Top / Rank to Bottom actions don't close when called from "." operations menu[512]
- Fixed: HJ-1120 Issue type drop-down does not contain default type[513]
- Fixed: HJ-1140 Structure causes error when displaying GH tab for issue in not Structure-enabled project[514]

7.27.9.3 Supported JIRA Versions

Structure 2.1 supports JIRA 5.0 — 5.2+.

7.27.9.4 Installation

The plugin may be installed from the Plugin Exchange[515] or from the **Administration | Plugins** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.9.5 Upgrading from a Previous Version

If you upgrade from version 1.7.1 or earlier, please read the upgrade section in Structure 2.0 Release Notes(see page 801).

Migration from version 2.0 to version 2.1 is pretty straightforward. Structure Backup is recommended as a safety measure.

---

506 https://jira.almworks.com/browse/HJ-1057
507 https://jira.almworks.com/browse/HJ-1069
508 https://jira.almworks.com/browse/HJ-888
509 https://jira.almworks.com/browse/HJ-1059
510 https://jira.almworks.com/browse/HJ-1060
511 https://jira.almworks.com/browse/HJ-1093
512 https://jira.almworks.com/browse/HJ-1100
513 https://jira.almworks.com/browse/HJ-1120
514 https://jira.almworks.com/browse/HJ-1140
515 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

If you're using GreenHopper 6.1 and the new epics functionality, please review the configuration of your synchronizers. You might want to delete the old synchronizers and install new GreenHopper synchronization that works with the new epics.

JIRA administrators are advised to contact the users who had installed GreenHopper synchronizers and advise them about the changes in Structure 2.1. You can find out which structures have synchronizers at the **Manage Structures** page.

Proper upgrade sequence:

1. Create backup of Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the whole `structure/` sub-directory under JIRA home while Structure is disabled. See Backing Up Structure(see page 486) for details.
2. Install the new version of the plugin.

   Note for early adopters: if you have a pre-release version of Structure 2.1 installed, you need to uninstall it first, and only then install the released version. All structure data will be kept.

3. Monitor `catalina.out` or `jira-application.log` for log messages from Structure.
4. Recreate GreenHopper synchronizers if needed, and advise the owners of the structures about the new GreenHopper synchronizer's functionality.

7.27.9.6 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 2.1[516].

7.27.9.7 Structure 2.1.1 Release Notes

18 March 2013

Structure 2.1.1 contains a critical fix for JIRA 5.2.8 and later versions.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[517]

1. Patch Release

This is a patch release based on version 2.1 and contains a workaround for compatibility problem in JIRA 5.2.8[518].

The upgrade is required if you use JIRA 5.2.8 or later; otherwise, the upgrade is optional.

1.1 Bugs Fixed

- Fixed: HJ-1182 Localized strings are not correctly resolved in JIRA 5.2.8[519]

516 https://jira.almworks.com/secure/StructureBoard.jspa?s=145
517 https://plugins.atlassian.com/plugin/details/34717
518 https://jira.atlassian.com/browse/JRA-32038
519 https://jira.almworks.com/browse/HJ-1182

2. Supported JIRA Versions

This version supports JIRA 5.0--5.2.8 and later.

This version does not support JIRA 6.0 EAP – support for 6.0 series will be added later.

7.27.9.8 API Changes in Structure 2.1

Minor API Release

API changes in Structure 2.1 are backward-compatible. All plugins compiled against previous 7.x version of the API should work without recompilation.

The new API version contains additional methods and a few minor improvements.

JIRA Version	New API Version
5.0 – 5.2	7.3.0

See Structure API Versions(see page 883) for full version information and downloads.

Java API: Backwards-Compatible Changes

- Several methods added to `RestoreOperation` interface for retrieving more information about problems that happened during restore.
- `SyncLogger.issues()` method for constructing debug messages about multiple issues.

7.27.10 Structure 2.0 Release Notes

19 November 2012

Structure 2.0 is a major release with six brand new features, support for JIRA 5.2 and lots of major improvements.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[520]

7.27.10.1 Version Highlights

- Views: persistent, shareable, reusable column configurations stored on the server.
- Template Structures and Bulk Issue Cloning: it is now possible to copy a whole structure and clone every issue it contains.
- Bulk Change: select multiple issues in Structure and click a button to open JIRA's bulk change wizard.
- Favorite Structures: quick access to few selected structures.
- Structure Tab in the GreenHopper's Issue Detail View: see and change selected issue's position in the hierarchy.

520 https://plugins.atlassian.com/plugin/details/34717

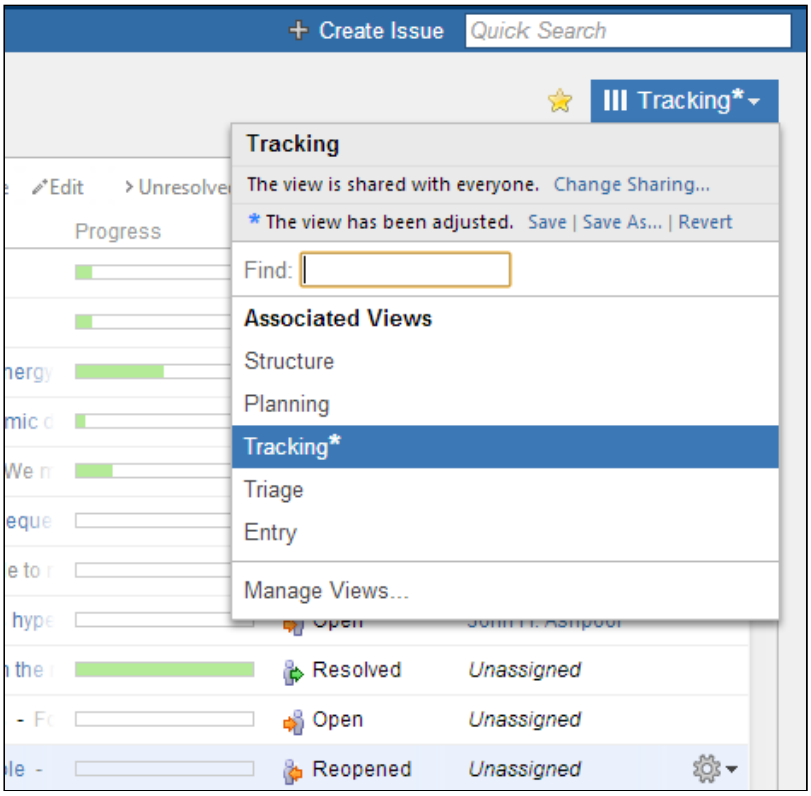
- Migrate Structures: JIRA administrators can now import structures from another JIRA instance.

This version also contains a lot of other notable improvements.

> ⚠ Structure Backup is absolutely necessary before the upgrade! Please read #Upgrading from a Previous Version(see page 808) below.

7.27.10.2 Changes in Detail

-
  - Views(see page 803)
  - Template Structures and Bulk Issue Cloning(see page 803)
  - Bulk Change(see page 804)
  - Favorite Structures(see page 804)
  - Structure Tab in GreenHopper Issue View(see page 804)
  - Structure Migration and Partial Import(see page 805)
  - Easier Structure Management(see page 805)
  - New Column Resizing and Automatic Widths(see page 805)
  - Dashboard and Confluence Gadget with Preset Columns(see page 806)
  - Reusing Columns Configuration from Issue Navigator(see page 806)
  - Revamped Multiple Issue Selection(see page 806)
  - Activity Streams Performance Improvements(see page 807)
  - Underlying Derby Database Upgrade(see page 807)
  - Other Improvements and Notable Bugs Fixed(see page 807)



Views

Views are named column configurations.

Now, when viewing a structure, you can quickly switch between different views and the columns in the grid will be quickly reconfigured. Views are stored on JIRA server, so you can create a new view and share it with your team.

It is also possible now to configure Structure gadget on JIRA dashboard, or in Confluence, to display a structure with specific columns.

Documentation: Configuring View(see page 336), Views Menu(see page 337), Managing Views(see page 341)

New keyboard shortcuts: **vv** (switch view), **vs** (save view) and a few others(see page 384).

Note: After upgrade, the earlier column configurations made by users and stored in browser cookies will be lost and replaced with the default view. You will need to reconfigure columns again, but this time you will be able to save the columns configuration as a view and share it between structures and with other users.

Template Structures and Bulk Issue Cloning

It's now possible to create a copy of a structure and clone every issue in that structure. The result is a complete duplicate of the structure and issues in it.

This lets you set up template structures or even template projects, and quickly create a new structure or project with preset issues and their hierarchy.

Cloning process is highly configurable, letting you to do intelligent copy of the template issues. And this feature plays well with Bulk Change feature, which later will allow you to further adjust newly created issues en masse.

Documentation: Copying Structure and Cloning Issues(see page 368), Template Structures and Projects(see page 374)

Bulk Change

You can now change multiple issues in structure using JIRA's Bulk Change wizard.

Select several issues in a structure using our brand new multi-selection feature (hit *Space* or click grey dot to select an issue). Then click Bulk Change button in the Structure toolbar, and it will take you to the standard JIRA's Bulk Change wizard, where you can choose to **Edit**, **Move**, **Transition** or **Delete** the selected issues.

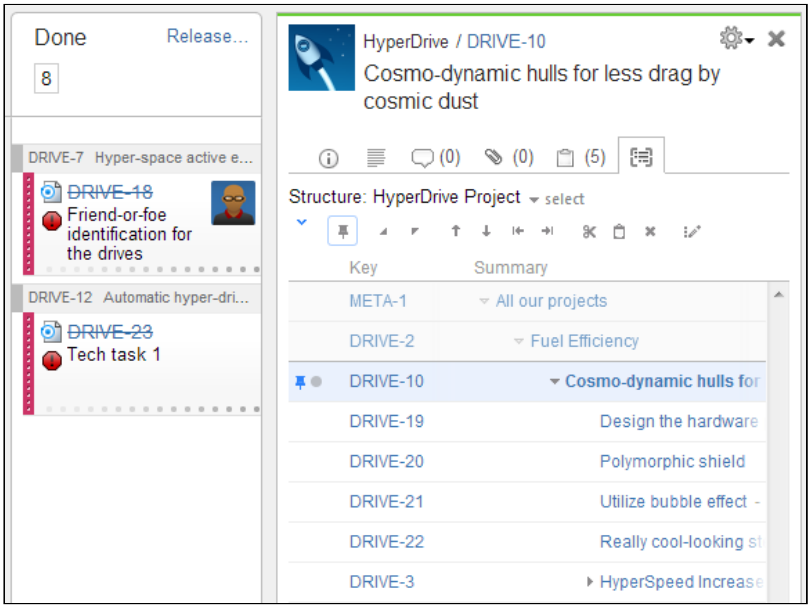
Documentation: Bulk Change(see page 120), Selecting Multiple Items(see page 101)

Favorite Structures

You can now mark a structure as your favorite, by clicking the familiar grey/yellow star near the structure name, in Manage Structures dialog or on the Structure Board.

Favorite structures are displayed in a separate section in the Structure top-level menu and they are grouped in a separate tab on the Manage Structures page.

Documentation: Favorite Structures(see page 357)



Structure Tab in GreenHopper Issue View

GreenHopper now displays *Structure* tab in the Issue Detail View, which allows you to quickly see the position of an issue in a structure.

Click an issue in a Rapid Board, and Details view appears, with *Structure* tab following the standard tabs. Structure tab displays the Structure widget, focused on the selected issue, much like on the issue page. You can select which structure to display, and you can Pin the issue to view only relevant part of the structure (parent issues and sub-issues) or Unpin to view the whole structure.

As there's usually not much space in the details view, Structure interface there is a bit scaled down, but almost fully functional, supporting inline editing and issue creation.

Structure tab is available for GreenHopper 6 or newer.

Documentation: Structure on Agile Boards(see page 91)

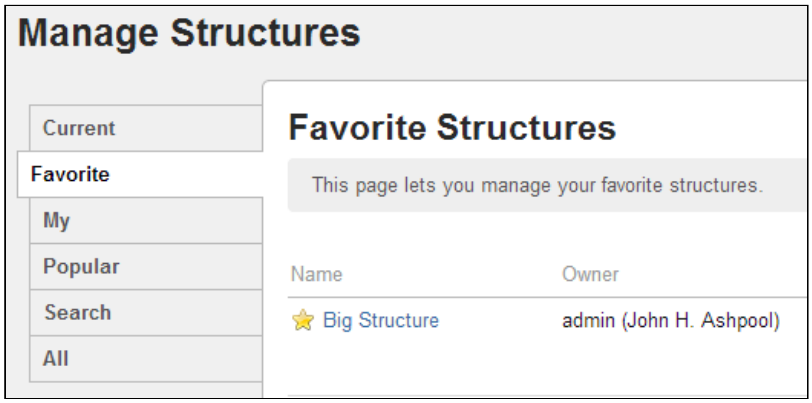
Structure Migration and Partial Import

JIRA administrators are now able to move structures from one instance of JIRA to another.

Until version 2.0, Structure supported only full backup and restore, and restore operation had to be performed on the same JIRA instance (because issues were identified by their numeric IDs). That was making it impossible to transfer structures to another JIRA after a project had been transferred via JIRA's Import Project feature.

In version 2.0, we introduce Migrate Structure tab in the Administrator's interface to Structure. It restores structures by identifying issues by their issue keys, and it also allows to pick which structures are to be restored.

Documentation: Migrating Structures(see page 488)



Easier Structure Management

The **Manage Structures** page now has got tabs and search, which let you quickly find the structure you need. It is especially handy if you've got a lot of structures in JIRA.

The tab that opens by default is **Current** tab, which lets you manage the structure you've just been working with.

Documentation: Managing Structures(see page 354), Locating a Structure(see page 356)

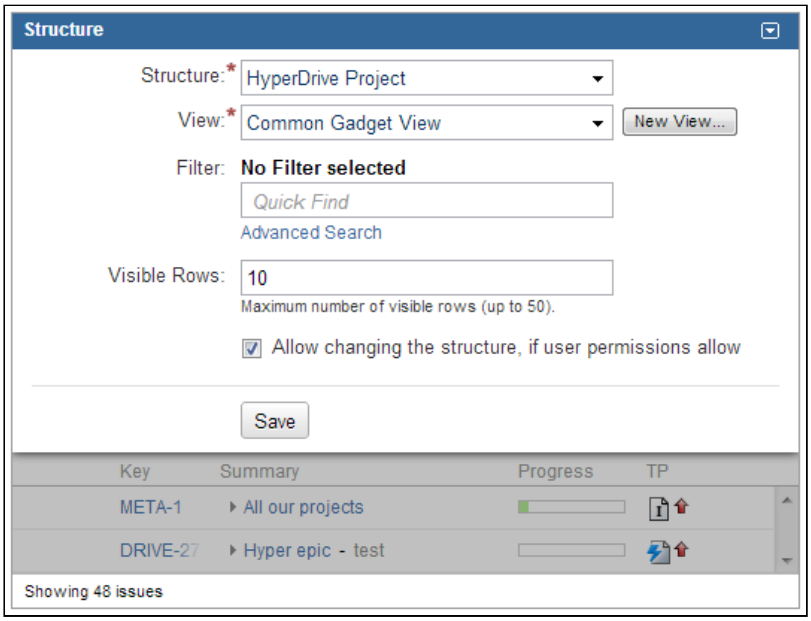
New Column Resizing and Automatic Widths

We have implemented a more convenient way to change column widths with the familiar "resizer" controls in the Structure grid header. But more importantly, the columns are now assuming the best width by default, based on the displayed data.

If you still need to change a column's width, drag the resizer responsible for that column (either to the left or to the right of the column header). In the proximity of the "best width", calculated for that column, the resizer snaps to the best position. You can alter the way resizer behaves by holding *Ctrl* or *Alt* keys while dragging the resizer.

Double-click a resizer or a column header to automatically set the best width for the column. Double-click *Summary* column or a new "resize all" button to set the best width for all columns.

Documentation: Customizing Columns(see page 331)



Dashboard and Confluence Gadget with Preset Columns

Structure dashboard gadget is fully integrated with the new Views feature. You can now decide which columns do all users see in the gadget when they open JIRA dashboard or a Confluence page.

When configuring a gadget, you can select a view – it defines which columns are displayed for all users. You can also create a new view on the spot, then add and rearrange columns the usual way, and save the changes into the view so the other users also see them.

Besides, we have given dashboard configuration panel a new look.

Documentation: Structure Gadget(see page 394)

Reusing Columns Configuration from Issue Navigator

When using Search in Structure, it's now possible quickly switch to the columns configured in the Issue Navigator.

Click **Columns** button on the search toolbar and the active view will temporarily switch to display the same columns you'd see in the Issue Navigator. When you turn **Columns** button off or when you close Search panel, view switches back to the one you were using.

Documentation: Using Issue Navigator Columns(see page 801)

Revamped Multiple Issue Selection

We have improved the way multiple selection works and looks, and we have made more Structure actions (like *Delete*) work with multiple selected issues.

Hit *Space* to select an issue, or click the grey circle at the left side of the issue row. Use *Shift+Arrows* to select a range of issues or *Ctrl+A* (*Command+A*) to select or deselect all issues. Hit *Escape* to clear selection. There are more keyboard shortcuts(see page 384) for the multi-selection.

Documentation: Selecting Multiple Items(see page 101)

Activity Streams Performance Improvements

We have optimized Structure's Activity Streams Provider, resolving critical performance issues for production systems with large structure history.

Sub-optimal implementation of Activity Streams (which is, basically, searching through structure change history) has been causing performance issues on JIRA instances where structure history had grown large. The activity stream wouldn't load in time and cause wasted CPU cycles and increased memory consumption. To make matters worse, JIRA does not allow to select activity streams providers for the "Activity" tab on the issue page, and for the user's and project's activity sections, so you could not turn off Structure activity streams there. The only remedy was to disable Activity Streams module in the Structure plugin.

We have taken great pains to fix this issue and ensure that Structure's activity streams are lightning fast, almost independent on the amount of structure history. The speed of search depends of course on the search conditions that the user specifies, but for typical conditions (and, more importantly, for the issue, project and user activity sections) the new activity streams is even faster than the standard JIRA's activity stream.

Underlying Derby Database Upgrade

Apache Derby embedded database, which runs Structure data behind the scenes, has been upgraded to the latest version, providing more stability and performance for Structure.

Other Improvements and Notable Bugs Fixed

- HJ-918 Limit the positive feedback between synchronizers[521]
- HJ-800 Make warnings logged by synchronizers more self-explanatory[522]
- Fixed: HJ-816 Double-escaped apostrophe on IE8[523]
- Fixed: HJ-967 Incorrect reaction to conflicting JIRA shortcuts like "g,i"[524]
- Fixed: HJ-886 Memory leak when enabling/disabling the plugin caused by Derby database incomplete shutdown[525]

7.27.10.3 Supported JIRA Versions

Structure 2.0 supports JIRA 4.4 — 5.2+. Note that there are separate downloadable files for JIRA 4.4 – 4.4.5 and for JIRA 5.0 – 5.2+.

This is the last version to support JIRA 4.4 series. Next minor release of the Structure plugin will support JIRA versions 5.0 and newer.

7.27.10.4 Changes for Developers

There are also a few goodies for the developers who integrate with Structure.

We have added a fully-functional `/structure` resource to the REST API, letting you manage structures remotely. Java API provides new managers, which are responsible for the new functionality in version 2.0.

521 https://jira.almworks.com/browse/HJ-918
522 https://jira.almworks.com/browse/HJ-800
523 https://jira.almworks.com/browse/HJ-816
524 https://jira.almworks.com/browse/HJ-967
525 https://jira.almworks.com/browse/HJ-886

For details, see API Changes in Structure 2.0(see page 809).

7.27.10.5 Installation

The plugin may be installed from the Plugin Exchange[526] or from the **Administration | Plugins** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.10.6 Upgrading from a Previous Version

Important! The upgrade makes irreversible changes to the Structure database. (JIRA's main database is not affected.)

Please back up Structure before upgrading!

It will not be possible to downgrade just by installing a previous version! See below for downgrade instructions.

The upgrade to version 2.0 involves changes in the Structure's database schema. The changes are made automatically, you only need to install the plugin.

Proper upgrade sequence:

1. Create backup of Structure data. You can either use **Administration | Structure | Structure Backup** menu or do a cold backup by copying the whole `structure/` sub-directory under JIRA home while Structure is disabled. See Backing Up Structure(see page 486) for details.
2. Verify that you have enough free disk space. If not sure that you have enough: a) see how much `structure/` sub-directory in JIRA home takes, b) make sure there's at least twice as much space free on that file system.
3. Install the new version of the plugin.

   Note for early adopters: if you have a pre-release version of Structure 2.0 installed, you need to uninstall it first, and only then install the released version. All structure data will be kept.

4. Monitor `catalina.out` or `jira-application.log` for log messages from Structure. Shortly after installation, Structure 2.0 should start a background migration process that will steadily transfer structure history from old schema to the new schema. Structure plugin can be used immediately after upgrade, but activity streams and structure history will be filled up gradually.

Downgrading

If for some reason you decide to downgrade to an earlier version of Structure:

1. Make sure you have the backup made earlier ready.
   - If you don't have the backup, but Structure 2.0 is operational, you can still use **Administration | Structure | Backup Structure** menu to create an XML backup, which will be readable by previous versions.
2. Uninstall Structure 2.0.
3. Manually delete or move away `structure/` sub-directory in JIRA home.
4. If you have a cold backup of `structure/` sub-directory, restore the directory contents from it.
5. Install previous Structure version.

526 https://marketplace.atlassian.com/plugins/com.almworks.jira.structure

6. If you didn't have a cold backup, but have an XML backup, use **Administration | Structure | Restore Structure** menu to restore data.
7. Let us know[527] about the reasons to downgrade, so we can address them.

7.27.10.7 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 2.0[528].

7.27.10.8 API Changes in Structure 2.0

Minor API Release

Although this is a major Structure release, the API changes are backward-compatible. All plugins compiled against previous 7.x version of the API (6.x for JIRA 4.4) should work without recompilation.

The new API version contains interfaces to some of the new functionality and accompanying data classes.

JIRA Version	New API Version
5.0 – 5.2	7.2.0
4.4.x	6.2.0

See Structure API Versions(see page 883) for full version information and downloads.

REST API: New `/structure` Resource

An entirely new REST resource, `/structure`, has been added to provide you with the ability to manage structures remotely. You can create, update, delete and query structures.

For details, see Structure Resource(see page 897) in the API Reference.

Java API: Backwards-Compatible Changes

`StructureViewManager` and `com.almworks.jira.structure.api.view` package

The new manager is responsible for Views(see page 336), a new type of entity introduced in Structure 2.0. You can create, retrieve, search and update views.

Related documentation: Managing Views(see page 341), Views Menu(see page 337)

`StructureFavoriteManager` interface

This manager is responsible for keeping track of favorite structures.

Related documentation: Favorite Structures(see page 357)

527 mailto:support@almworks.com
528 https://jira.almworks.com/secure/StructureBoard.jspa?s=140

`StructureListener` contract revised

After careful examination we discovered that `StructureListener`, provided in the earlier versions of the API, may have an undocumented behavior – namely, the order of notifications is not guaranteed to be the same as the order of mutations. If you have changes A and then B, you might be notified about B first, and then about A.

For some listeners this might break things, so we have a) documented this behavior, and b) introduced another type of listener, `SequentialStructureListener`, which can be used to ensure that you get notifications in the right order. However, it comes with a price of increased complexity – see the description of the listener interface.

Other Changes

- `StructureJobManager.execute()` method added to execute a job, bypassing the common single-thread queue.
- `SyncLogger` utility class added to make it simple for synchronizer authors to provide many details in the log messages from synchronizers.
- Additional features in `La` class and other minor changes.

7.27.11 Structure 1.7 Release Notes

11 July 2012

Structure 1.7 introduces support for JIRA 5.1, provides more options in the Filter synchronizer configuration, and adds a number of smaller improvements.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[529]

7.27.11.1 Version Highlights

- Support for JIRA 5.1 (with support for the latest GreenHopper versions up to 5.10.x)
- More flexible Filter synchronizer, with JQL Query and Allow Move options
- Page reload not required on workflow transitions and most other JIRA actions, resulting in much smoother user experience

7.27.11.2 Changes in Detail

Support for JIRA 5.1

JIRA 5.1 has been released a couple of days ago and we're happy to announce that Structure 1.7 is fully compatible with the latest JIRA version.

The biggest change in 5.1, from the user interface perspective, is the new issue page with the inline editing of the field values. We did our best to make Structure play well with the new functionality in JIRA. Now when you change issue values in-line on the issue page, they are immediately updated in the Structure widget, and vice versa.

529 https://marketplace.atlassian.com/plugin/details/34717

Filter Synchronizer Improvements

This version of Structure has an upgraded Filter synchronizer, which allows you to use any JQL query instead of a pre-existing Saved Filter. This makes it much easier to set up one or more filter synchronizers on a structure.

If you configure Filter synchronizer to add matching issues under a specific parent issue, there's now a new option for you, called Allow Move. When this option is turned on, and a matching issue is already in the structure – but at a different location, it will be moved under the specified parent issue. When this option is off, issues that are already added to the structure will never be moved. (This is how Filter synchronizer used to work before.)

Documentation: Filter Synchronizer(see page 524)

Less Page Reloads

Applicable to JIRA 5.0 - 5.1.x

In Structure 1.7, when you do a workflow transition (such as *Resolve Issue*) on the Structure Board, the change is uploaded and you can continue working on the same page – without waiting until it reloads. Most other JIRA actions that are available from the "Cog" drop-down or from the Operations dialog (triggered by the "." shortcut) also do not require page reload.

This seemingly small improvement really changes the feel of the application when you start working on a list of issues – especially if you like using the keyboard.

Other Improvements and Notable Bugs Fixed

Among other improvements:

- The adjusted Time Tracking section on the issue page now remembers the position of the "Use Structure" switch (now called "Include sub-issues".) (HJ-746[530])
- Anonymous users are no longer allowed to install synchronizers or run Import/Export, even if they have Control permission on a structure. (HJ-711[531])
- JIRA Agile (GreenHopper) Synchronizer(see page 810) can now ignore Fix Version if synchronizing with the latest GreenHopper versions. The synchronizer used to check that affected issues have empty or non-released Fix Versions, which may now be obsolete when using Rapid Boards. (HJ-769[532])
- Structure Plugin will now fail to start and will attract the administrator's attention if the underlying database is read-only (so the problem can be solved before it escalates.) (HJ-712[533])
- Bug fixed: HJ-751 Ctrl+Click and Cmd+Click don't work on link inside Structure Widget as expected (open link in a new tab)[534]

7.27.11.3 Supported JIRA Versions

This version supports JIRA versions 4.4 – 5.1.x. There are two separate downloadable files: one for JIRA 4.4.x, and one for JIRA 5.0 – 5.1.x.

Support for JIRA 4.3 is dropped starting with this version. The last working version for JIRA 4.3 is Structure 1.6.0. You can download older versions of Structure from Download Archive(see page 979).

530 https://jira.almworks.com/browse/HJ-746
531 https://jira.almworks.com/browse/HJ-711
532 https://jira.almworks.com/browse/HJ-769
533 https://jira.almworks.com/browse/HJ-712
534 https://jira.almworks.com/browse/HJ-751

7.27.11.4 Installation

The plugin may be installed from the Atlassian Marketplace[535] or from the **Administration | Plugins** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.11.5 Upgrading from a Previous Version

Migration to version 1.7 from version 1.6 is pretty straightforward and does not require unusual actions. Structure Backup is recommended as a safety measure.

Filter synchronizer configuration will be upgraded to preserve the functionality that was in place in the previous version. (That is, Allow Move setting(see page 524) will be **Off** by default.)

When upgrading from a version earlier than 1.6, please consult Upgrading section in all relevant (HIDDEN) Release Notes (Structure 1-2)(see page 761).

7.27.11.6 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 1.7[536].

7.27.11.7 Structure 1.7.1 Release Notes

> **7 August 2012**
>
> Structure 1.7.1 contains a compatibility fix for the latest GreenHopper.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[537]

1. Patch Release

This is a patch release based on version 1.7 and contains a single fix for a compatibility problem with GreenHopper 5.10.6.

1.1 Bugs Fixed

- Fixed: HJ-825 Incompatibility introduced by GreenHopper API change in GH 5.10.6[538]

2. Supported JIRA Versions

This version is released specifically for JIRA versions 5.0--5.1, there is no backport for JIRA 4.4 because GreenHopper 5.10.6 is not compatible with JIRA 4.4.

535 https://marketplace.atlassian.com/plugin/details/34717
536 https://jira.almworks.com/secure/IssueNavigator.jspa?mode=hide&requestId=10570
537 https://plugins.atlassian.com/plugin/details/34717
538 https://jira.almworks.com/browse/HJ-825

7.27.11.8 API Changes in Structure 1.7

Minor API Release

The changes in the API in this release are really minor. We had to bump minor version of the API because we've added another constant to the `StructureError` enum. This is literally all changes - if you have your code working under API 6.0.x or 7.0.x, it will surely work with 6.1.x and 7.1.x, respectively.

JIRA Version	Structure Version	New API Version
4.4.x	1.7.0.jira44	6.1.0
5.0 – 5.1.x	1.7.0	7.1.0

See Structure API Versions(see page 883) for full version information and downloads.

7.27.12 Structure 1.6 Release Notes

12 May 2012

Structure 1.6 introduces a number of major improvements, including much more powerful Links synchronizer, more options when creating a new issue, dashboard gadget filtering and other improvements.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[539]

7.27.12.1 Version Highlights

- Links Synchronizer rewritten to be much more powerful and consistent
- It's now possible to select Project and Issue Type when creating new issues *(on JIRA 4.4.3 and later versions only)*
- Dashboard gadget now support filtering
- Automatic selection and other convenience improvements on the Issue Page and Project Pages

7.27.12.2 Changes in Detail

Links Synchronizer Taken to the Next Level

One of the most popular synchronizers, Links Synchronizer, was rewritten almost from scratch to address a number of issues raised in our JIRA. The new synchronizer is much more powerful and consistent in operation.

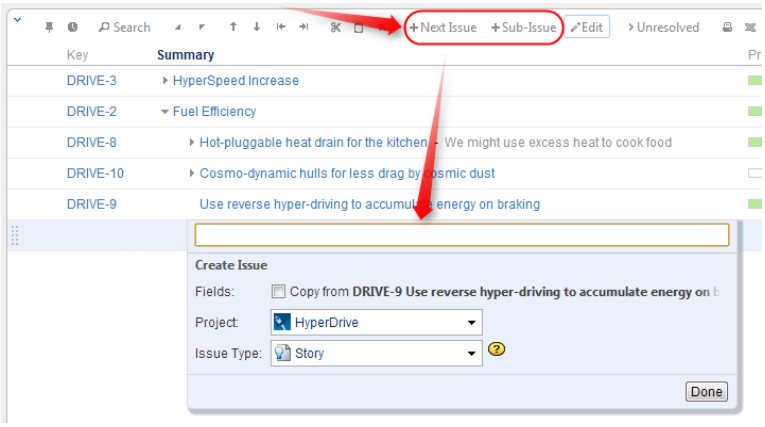
Features of the new synchronizer:

539 https://plugins.atlassian.com/34717

- It can track links creation / removal and reflect that in the structure;
- It can add issues to the structure that have incoming links from other issues that are already in the structure. (Effectively expanding the structure based on the issues that are already there, but not touching the issues that are not reachable from the structure through links.)
- It can work on a sub-set of issues, defined through JQL / Saved Filter (different filters possible for parent and child issues).
- It is now possible to create multiple Links Synchronizers on the same structure that would operate on different levels using different link types (the synchronizers' scope should be separated via filtering).
- It is now possible to enforce that structure changes that contradict the links are reversed - or vice versa.

Upgrade note: if you have Links synchronizer installed, the upgrade should run smoothly. The new synchronizer should convert the old settings to the new settings. However, we advise that you review the possibilities with the new synchronizer options and probably re-create the synchronizers.

Documentation: Links Synchronizer(see page 528)



Selecting Project and Issue Type for New Issues

This feature is not available on JIRA 4.3. and on JIRA 4.4 – 4.4.2.*

Until now, creating new issues inside Structure widget actually meant to clone an existing issue and change its summary. That also meant that the Project and Issue Type of the new issue would be the same as the previously selected issue.

In this version, we have added the possibility to create issue "from scratch", in the same way they are normally created. The **Summary** field editor for the new issues now displays controls that let you switch from the old "clone" mode to the new "from scratch" mode and back, and select Project and Issue Type.

If you're a keyboard fan, try hitting **Enter** and **Ctrl+Enter** after you start creating a new issue - but before you write text in the summary field.

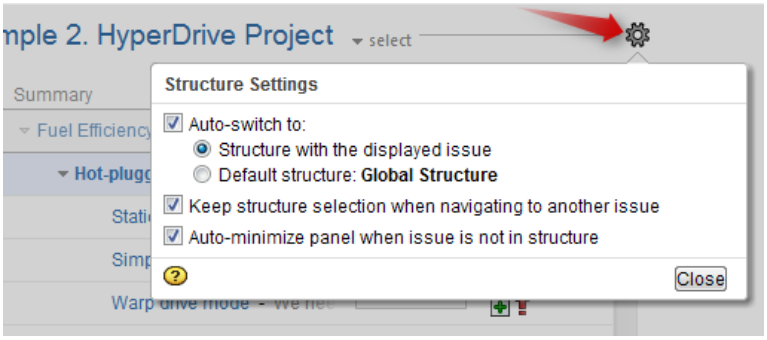
Documentation: Creating New Issues(see page 110)

Filtering Structure on Dashboard Gadget

Dashboard gadget configuration now supports filtering. You can edit a gadget and select a Saved Filter - the structure will be filtered in the same way it's filtered on the Structure Board when **Filter** button is on.

As a side effect of this feature, a feature of its own: filtering is now real-time when turned on in the structure widget, that is, if an issue is changed and it no longer satisfies the filter, it is hidden. If an issue changes and now it satisfies the filter, it is shown.

Documentation: Structure Gadget(see page 394), Searching and Filtering(see page 813)



Issue Page: Automatic Structure Selection and More

We have added a number of options, configurable per-user, to provide a better experience with Structure on the issue page:

- **Auto-switch** settings let you have Structure section automatically switch to the structure that contains the displayed issue, or to the default structure, configurable per-project.
- Another option, **keep structure when navigating**, allows to override the auto-switch when you click on an issue within the structure widget.
- **Auto-minimize** option forces Structure section to collapse when the displayed issue is not present in the selected structure.

Documentation: Structure Options for the Issue Page(see page 88), Default Structure(see page 357), Changing Structure Defaults(see page 481)

Project/Component/Version Page: Automatic Structure Selection

Similar to the options on the issue page, Structure tabs on the Project, Component and Version pages now allow to turn on **Auto-switch** setting, which would have the tab display the default structure (configurable per-project) when the page is loaded.

> The auto-switch settings for the Issue Page and Project Pages are not *enforcements* – the user can still switch to another structure. It's a matter of convenience and avoiding confusion.

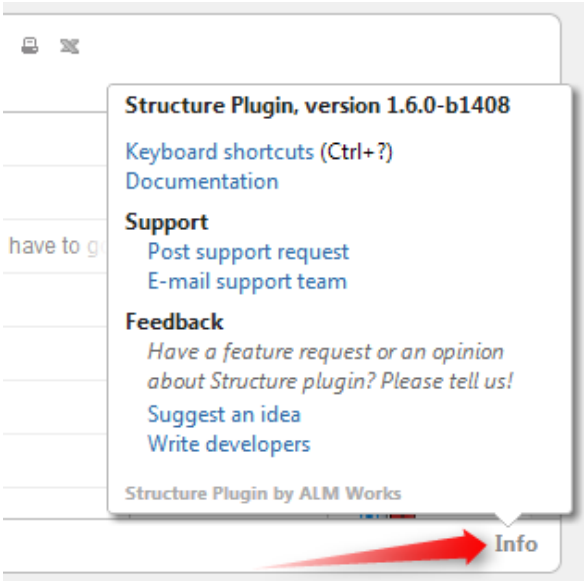
Documentation: Structure on the Project Page(see page 90), Default Structure(see page 357), Changing Structure Defaults(see page 481)

Simplified User Interface in One-Structure Case

Now when there's only one structure in the system (or if the user has access to only one structure), the issue page and project pages do not display structure selectors and the name of the selected structure. The section is just named "Structure".

Structure menu also displays only one modest **Open Structure** item (along with **Manage Structure** and **Create Structure** when applicable).

Documentation: Structure Menu(see page 74)



"Info" Footer Link

We have removed **Help**, **Feedback** and **Keyboard** links from the Structure Widget's footer and replaced them with a single **Info** link, which shows the version of Structure, link to keyboard shortcuts cheat sheet and other useful links.

Documentation: Getting Help(see page 403)

Noteworthy Bugs Fixed

- HJ-680 Upgrading Universal Plugin Manager from 2.0.4 to 2.1.1 kills Jira with the issue appearing to start in Structure[540]
- HJ-688 Saved Filter synchronizer does not remove parent issue with sub-issues[541]
- HJ-699 If an issue is deleted, it is still kept in the structure and skews Progress%[542]

7.27.12.3 API Changes

If you're developing your own plugin that uses Structure API, make sure you go through the API Changes in Structure 1.6(see page 817) to see if you need to adjust your code. We had to bump the major API version in this release due to some incompatible changes.

7.27.12.4 Supported JIRA Versions

This version supports JIRA versions 4.3 – 5.0.x. There are separate downloadable files for JIRA 5.0.x, 4.4.x and 4.3.x. We're currently working on support for early 5.1 versions.

> ⚠ JIRA 5.1 is coming out soon, which means that a new version of Structure, which supports JIRA 5.1, will no longer support JIRA 4.3.

540 http://jira.almworks.com/browse/HJ-680
541 http://jira.almworks.com/browse/HJ-688
542 http://jira.almworks.com/browse/HJ-699

7.27.12.5 Installation

The plugin may be installed from the Plugin Exchange[543] or from the **Administration | Plugins** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.12.6 Upgrading from a Previous Version

Migration to version 1.6 from earlier versions is pretty straightforward and does not require unusual actions. Manual Structure Backup(see page 486) is highly recommended as a safety measure.

After the upgrade, please check your links synchronizers - you might want to re-create them using more flexible options of the new Links Synchronizer(see page 528).

When upgrading from a version earlier than 1.5.*, please consult Upgrading section in all relevant (HIDDEN) Release Notes (Structure 1-2)(see page 761).

7.27.12.7 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 1.6[544].

7.27.12.8 API Changes in Structure 1.6

Major API Release

We have to make a number of incompatible changes in API when working on Structure 1.6. If you're using the API, please check these change notes and see if you need recompile your code.

JIRA Version	New API Version
4.3.x	6.0.0
4.4.x	6.0.1
5.0.x and later	7.0.0

See Structure API Versions(see page 883) for full version information and downloads.

543 https://plugins.atlassian.com/34717

544 http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.6%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true

Incompatible Changes

`IncrementalSyncData` changed

It is now an interface, so you cannot instantiate it. The old methods are remaining the same, but there's a new method `getSyncEvents`, which allows you to get a list of events (not issues) that should be synchronized. This allows for a more fine-grained synchronization algorithms.

The class representing the event - `SyncEvent` - is part of this change.

`IssueListener` changed

`IssueListener.onIssueChanged()` now accepts `JiraChangeEvent` and not `long` as before. If you've used our `IssueEventBridge` to listen for events, you need to update your listener. On the same note, `IssueEventBridge` methods for reporting the events changed as well - see the API.

The new classes `JiraChangeEvent`, `IssueChangeEvent` and `LinkChangeEvent` are part of this change.

`StructureListener.StructureChanges` is now an interface

You can no longer instantiate this class. We hope you didn't need to, before.

`ModuleStopListener` utility class removed

It wasn't working well as it could miss component destruction in some events. If you were using it, consider switching to `DisposableBean`.

`AbstractSynchronizer.issueDebug()` is not a static method

It used to be an instance method, but we had to convert it to a static one. If you had used it, recompilation is required.

Compatible Changes

Support for plugin defaults in `StructureConfiguration` interface

A number of methods have been added to `StructureConfiguration` to support persistence of the new settings – default structures and UI settings such as auto-switch.

Documentation for the features: Changing Structure Defaults(see page 481), Structure Options for the Issue Page(see page 88)

`StructureManager` new method: `getSingleViewableStructureId(User)`

The new `getSingleViewableStructureId(User)` in `StructureManager` allows to detect that the user can only see a single structure, and this can be used to simplify the user interface (for example, hide the structure selector).

`Forest` new methods

- Method `getPathForIndex(int)` to get the path to an issue, similar to `getPath(long)`
- Method `foldUpwards(ForestParentChildrenClosure)` for efficient traversal and folding of the forest from bottom to top, comes with auxiliary classes `ForestParentChildrenClosure` and others.

`StructureListener.StructureChanges.getForestOps()`

Class `StructureListener.StructureChanges` now contains method `getForestOps()` that allows you to inspect the events that have happened in detail. This is especially useful in synchronizers.

`SyncController` additional methods

`SyncController` now has another variant of `incrementalSyncRequired` method that accepts a `JiraChangeEvent`.

Safer `Structure` methods

Methods `getName()` and `getDescription()` now return non-null values always.

7.27.13 Structure 1.5 Release Notes

17 February 2012

Structure 1.5 adds support for JIRA 5 and the latest GreenHopper versions.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[545]

7.27.13.1 Version Highlights

- Support for the upcoming JIRA 5.
- Support for GreenHopper 5.8.6 and the upcoming GreenHopper 5.9.

7.27.13.2 Changes in Detail

JIRA 5 Support

JIRA 5 is coming out soon, and Structure 1.5 is the version for JIRA 5 that will be compatible at launch. Due to massive changes in JIRA 5, support for it is the biggest change in this Structure version.

Support for JIRA 5 has been tested with the latest Release Candidate 7.

Support for New GreenHopper Versions

GreenHopper 5.8.6 has a different Ranking API, so GreenHopper synchronizer in the older versions of Structure will not be able to change Rank field in the newest GreenHopper version.

Structure 1.5 introduces support for GreenHopper 5.8.6 and for GreenHopper 5.9 (tested against the latest Release Candidate). Structure still supports older versions of GreenHopper.

545 https://plugins.atlassian.com/plugin/details/34717

Change to the Operations Dialog Keyboard Shortcut on the Issue Page

The "." (dot) keyboard shortcut brings up JIRA's Operations dialog, which allows you to quickly apply an action to the selected issue.

Prior to Structure 1.5, using this keyboard shortcut on the Issue Page would open Operations dialog for the issue currently selected in Structure. Starting from Structure 1.5, pressing "." on the Issue Page always brings up Operations dialog for the issue being viewed, regardless of the issue selected in Structure. This is consistent with other JIRA keyboard shortcuts ("m", "l" and so on), that work in the same manner on the Issue Page.

Structure widget on other pages is not affected by this change.

Notable Bugs Fixed

- Bug fixed: HJ-591 Open Structure section in issue view breaks labels field[546]

7.27.13.3 Supported JIRA Versions

This version supports JIRA versions 4.3 – 5.0.x. There are separate downloadable files for JIRA 5.0.x, 4.4.x and 4.3.x.

> Support for JIRA 4.2 is dropped starting with this version. The last working version for JIRA 4.2 is Structure 1.4.1. You can download older versions of Structure from Download Archive(see page 979).

7.27.13.4 Installation

The plugin may be installed from the Plugin Exchange[547] or from the **Administration | Plugins** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.13.5 Upgrading from a Previous Version

Migration to version 1.5 from version 1.4.x is pretty straightforward and does not require unusual actions. Structure Backup is recommended as a safety measure.

When upgrading from a version earlier than 1.4, please consult Upgrading section in all relevant (HIDDEN) Release Notes (Structure 1-2)(see page 761).

7.27.13.6 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 1.5[548].

546 http://jira.almworks.com/browse/HJ-591
547 https://plugins.atlassian.com/plugin/details/34717
548 http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.5%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true

7.27.13.7 Structure 1.5.1 Release Notes

22 January 2012

Structure 1.5.1 contains a few minor bug fixes.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[549]

1. Patch Release

This is a patch release based on version 1.5 and contains fixes for a couple of bugs, discovered in version 1.5 for JIRA 5.

1.1 Bugs Fixed

- Fixed: HJ-658 Assignee in-place editor is 0px wide when the user does not have "browse users" permissions[550]
- Fixed: HJ-657 (JIRA 5) Expand/Collapse state of the Structure section on the Issue Page is not remembered[551]

2. Supported JIRA Versions

This version is released specifically for JIRA 5.0, there are no backports for JIRA 4.3 and 4.4. The next release will contain all backports and support JIRA versions 4.3 – 5.0.x.

7.27.13.8 Structure 1.5.2 Release Notes

16 March 2012

Structure 1.5.2 contains a number of bug fixes, including fixes for a couple major issues.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[552]

1. Patch Release

This is a patch release based on version 1.5 and contains bug fixes. For plugin developers, it also introduces `structure.widget` web-resource context.

549 https://plugins.atlassian.com/plugin/details/34717
550 http://jira.almworks.com/browse/HJ-658
551 http://jira.almworks.com/browse/HJ-657
552 https://plugins.atlassian.com/plugin/details/34717

1.1 `structure.widget` Web Resource Context

Plugin Developers: It's now possible to include your web resources on every page with the Structure widget, except dashboard gadget. That will make it possible to provide the necessary resources for the editors of your custom fields.

See details in the Developer Documentation(see page 879).

1.2 Bugs Fixed

- Fixed: HJ-671 Links import cycles in some cases, blocking structure update[553]
- Fixed: HJ-659 Double-click in the inline text editor should work as expected (select a word), not be blocked[554]
- Fixed: HJ-660 Unexpected nulls in structure permission rules may cause NPE[555]

Full list: issues resolved in Structure version 1.5.2[556]

2. Supported JIRA Versions

This version supports JIRA versions 4.3 – 5.0.x.

3. Upgrading

Upgrade to 1.5.2 is recommended, since the issues may affect general functionality of the plugin.

Upgrade from version 1.5 or 1.5.1 is straightforward and does not require special actions. For upgrading from earlier versions, please read the related (HIDDEN) Release Notes (Structure 1-2)(see page 761).

7.27.13.9 API Changes in Structure 1.5

Major API Release

Structure 1.5 comes with a set of API updates. The major changes are in the API for JIRA 5 compatible version.

JIRA Version	New API Version
4.3.x	4.0.2
4.4.x	4.0.3
5.0.x and later	5.0.0

553 http://jira.almworks.com/browse/HJ-671
554 http://jira.almworks.com/browse/HJ-659
555 http://jira.almworks.com/browse/HJ-660
556 http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.5.2%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true

See Structure API Versions(see page 883) for full version information and downloads.

Incompatible Changes

The only incompatible change in API 5.0.0 is that `StructureSynchronizerModuleDescriptor` class now extends a different base class. Theoretically this can break compatibility, but it's highly unlikely.

Compatible Changes

Utility Updates

`JiraFunc` class now contains more functions for operating on JIRA entities.

7.27.14 Structure 1.4 Release Notes

8 January 2012

Structure 1.4 brings forward three new major features - Excel export, Structure History with Activity Streams and Editing Issues On-the-Grid. This release also contains other new features and improvements.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[557]

7.27.14.1 Version Highlights

- Editing Issues On-the-Grid: you can now change values of most issue fields right in the Structure widget.
- Structure History: all changes to structures are now recorded and can be inspected with the Structure widget's history panel.
- Activity Stream: you can now display changes to structures along with other changes in the Activity Stream dashboard gadget.
- Export to Excel: the structure and issue fields can now be exported onto a neatly organized XLS file.

This version contains a number of other new features and improvements.

Structure Backup is strongly recommended before the upgrade.

557 https://plugins.atlassian.com/plugin/details/34717

7.27.14.2 Changes in Detail

Editing Issues On-the-Grid

Featured in the Video

Double-click a value in the Structure widget to edit it! Most fields that are editable through Edit Issue page can now be edited in Structure, without leaving for another web page.

Quite a few keyboard shortcuts have been added to allow quick editing without touching the mouse.

Documentation: Editing Issues(see page 114)

Change History & Activity Stream

Featured in the Video

All changes made to structures are now recorded, including changes made by synchronizers(see page 514). The change history can be inspected in the Structure widget by turning on History Bar.

On JIRA 4.4 and later versions, structure changes are also published as an activity stream and can be shown by the Activity Stream gadget.

Documentation: Viewing History of a Structure(see page 375), Structure Activity Stream(see page 381)

Export to Excel

Featured in the Video

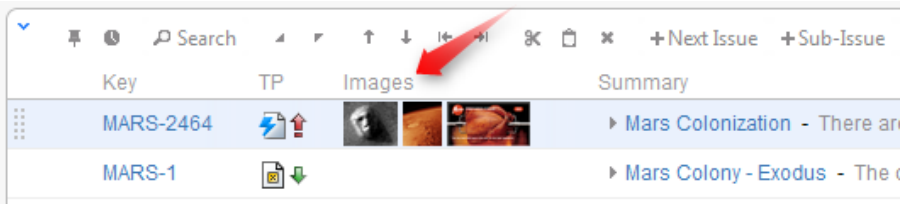
It's now possible to open the same structure, as you see it, in Excel – with a single click Structure plugin exports the structure as a neatly formatted XLS file.

Documentation: Exporting Structure to XLS (Excel)(see page 378)

Aggregated Time Tracking Columns

We have added columns that display the aggregated Time Spent, Remaining Estimate and Original Estimate for the issue and all its sub-issues.

Documentation: Time Sum Columns(see page 823)



Images Column

We have improved Images column, which now displays small thumbnails of all attached images. When you click on a thumbnail, a full-size image is shown in a pop-up dialog.

This column has special support for keyboard: press **i,i** ("i" twice) to show the first image of the selected issue, and use arrows to scroll through images.

Documentation: Images Column(see page 314)

Performance Improvements

We continually work on optimizing plugin performance both on the server side and on the browser side. In this version, we have eliminated lock contention on the server that could have caused trouble when Structure was used extensively by lots of users. We have also added caching to the structure access level checks.

Keyboard Shortcuts Reference Pages

We have added keyboard shortcuts reference to the documentation – see Keyboard Shortcuts (PC)(see page 385), Keyboard Shortcuts (Mac)(see page 389).

Other Improvements and Notable Bugs Fixed

- Fixed: HJ-499 Polling initiated by Structure on the view issue page prevents session timeout[558]
- Fixed: HJ-510 Enable GreenHopper Synchronizer to work with the Epic/Theme custom fields that are enabled only for specific issue types[559]

7.27.14.3 Supported JIRA Versions

This version of Structure supports JIRA 4.2 — 4.4.x. Note that there are separate downloadable files for JIRA 4.4.x, 4.3.x and 4.2.x.

An additional version that supports JIRA 5 Release Candidates will be delivered soon.

Activity Streams are supported only on JIRA 4.4 and later versions.

558 http://jira.almworks.com/browse/HJ-499
559 http://jira.almworks.com/browse/HJ-510

⚠ This is the last version to support JIRA 4.2.x. Next versions of the Structure plugin will support JIRA 4.3 — 5.0.x.

7.27.14.4 API Changes

This version of Structure introduced some changes to the Structure API that might break functionality of dependent plugins. See API Changes in Structure 1.4(see page 827) for details.

7.27.14.5 Installation

The plugin may be installed from the Plugin Exchange[560] or from the **Administration | Plugins** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.14.6 Upgrading from a Previous Version

Migrating to version 1.4 from an earlier version involves automatic upgrade of the underlying database data, so it's a good idea to make database backup. If you decide to downgrade back to an earlier version after upgrade, you may need to clear the database and restore Structure from backup.

7.27.14.7 Important Changes to Restoring Structure from Backup

Starting from this version, Structure plugin requires full restart (disable and enable) when Structure data is restored from backup. On JIRA versions 4.4 and later, after Restore operation has finished, the plugin is disabled and re-enabled automatically, so the full functionality is available again to the users.

On JIRA 4.2 and 4.3, Structure is not able to perform full restart, so after the restore Structure is temporarily disabled, and JIRA restart is required (for the same reason JIRA restart is required after installation of Structure).

7.27.14.8 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 1.4[561].

7.27.14.9 Structure 1.4.1 Release Notes

ⓘ **16 January 2012**

Structure 1.4.1 contains a couple of bug fixes.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[562]

560 https://plugins.atlassian.com/plugin/details/34717
561 http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.4%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true
562 https://plugins.atlassian.com/plugin/details/34717

1. Patch Release

This is a patch release based on version 1.4 and contains fixes to a couple of bugs.

1.1 Bugs Fixed

- Fixed: HJ-583 Server exception when creating a new issue that does not have Priority (NPE at CreateIssueOp.java:185)[563]
- Fixed: HJ-584 Keyboard shortcut "Enter" for starting a new issue does not work more than one time in Opera 11[564]

2. Supported JIRA Versions

This version supports JIRA versions 4.2 – 4.4.x.

3. Upgrading

Upgrade to 1.4.1 is optional and recommended if issues without Priority set are possible in your JIRA.

Upgrade from version 1.4 is straightforward and does not require special actions. For upgrading from earlier versions, please read the related (HIDDEN) Release Notes (Structure 1-2)(see page 761).

7.27.14.10 API Changes in Structure 1.4

Major API Release

Structure 1.4 comes with a major upgrade of the API, which might break compatibility with the plugins using older API.

JIRA Version	New API Version
4.2.x	3.0.0
4.3.x and later	4.0.0

See Structure API Versions(see page 883) for full version information and downloads.

Incompatible Changes

The following changes may break compatibility of older plugins. Please check thoroughly.

Integers library upgraded to version 0.73

New version of Integers library[565] is used in this API, which is not backwards-compatible with the older version 0.23.

563 http://jira.almworks.com/browse/HJ-583
564 http://jira.almworks.com/browse/HJ-584
565 http://code.google.com/p/integers/

Most classes and interfaces remain the same, but the iterators have changed. To make your code compatible with the new version, search your code for using methods `LongIterator.next()` and `IntIterator.next()` and replace them with `LongIterator.nextValue()` and `IntIterator.nextValue()` respectively.

`ForestOp` changed

If you're using this class (when analyzing incremental update of a forest), you need to revisit the documentation for ForestOp[566] and its sub-classes.

Compatible Changes

`BackupOperation`

`BackupOperation` now has method `setBackupHistory()` to govern whether to backup history of all structures.

`Forest`

`Forest` interface now has a few methods added. See javadoc[567] for a full method list.

7.27.15 Structure 1.3 Release Notes

24 October 2011

Structure 1.3 introduces the Structure API for integration with other plugins, and contains important performance improvements a number of minor updates.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[568]

7.27.15.1 Version Highlights

- Structure API that allows other plugins and remote applications to work with Structure.
- Big performance improvements for structures with many issues.
- Compatibility with the latest JIRA 4.4.3 and GreenHopper 5.8 releases.

Important Upgrade Notes

Structure Backup is strongly recommended before the upgrade.

Also, after upgrade you may need to re-create GreenHopper synchronizer configurations (if using GreenHopper 5.8 or later) and revisit Sub-tasks synchronizer configurations - please read below.

566 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestOp.html
567 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/Forest.html
568 https://plugins.atlassian.com/plugin/details/34717

7.27.15.2 Changes in Detail

Structure API

Structure API is the biggest thing that we've worked on since version 0.4, and as of Structure version 1.3 (API versions 1.0.0 and 2.0.0), it's a big window into the the structure data that lets other JIRA plugins and remote applications integrate with structure, read or update hierarchical issue lists and extend Structure functionality.

If you are a developer, take a look at our Structure Developer's Guide(see page 842).

Performance Improvements

We're constantly improving performance of our plugin, which is especially important if you have lots of issues in a single structure. In this version, we've made three different performance improvements which can lower down the load time of a large structure three to ten times, depending on the circumstances.

Support for GreenHopper 5.8

GreenHopper 5.8 and 5.8.1, released last week, replaced the old Rank field with new Global Rank field. Now the Global Rank in GreenHopper allows to prioritize issues across multiple projects on the Rapid Board.

Updated version of the GreenHopper synchronizer in Structure 1.3 supports the new Global Rank too, allowing you to select multiple projects in configuration.

Documentation: JIRA Agile (GreenHopper) Synchronizer(see page 828)

Sub-tasks Synchronizer Changes

Sub-tasks synchronizer logic has been made less confusing. Now when the synchronizer "sees" a sub-task at a wrong position in the structure, it will move it to the right position under its parent task. (The synchronizer used to move a sub-task only if it wasn't in the structure or was at the top level, but not under another parent issue.)

Documentation: Sub-Tasks Synchronizer(see page 524)

Other Improvements and Notable Bugs Fixed

- Bug fixed: HJ-500 Warning about losing input text when navigating away from an Issue Page that has Structure section with Search turned on[569]
- Bug fixed: HJ-495 Structure plugin over-rides submit keyboard shortcut when editing an issue[570]

7.27.15.3 Supported JIRA Versions

This version of Structure supports JIRA 4.2 — 4.4.x. Note that there are separate downloadable files for JIRA 4.4.x, 4.3.x and 4.2.x.

569 http://jira.almworks.com/browse/HJ-500
570 http://jira.almworks.com/browse/HJ-495

7.27.15.4 Installation

The plugin may be installed from the Plugin Exchange[571] or from the **Administration | Plugins** menu. Further information is available in the Administrator's Guide(see page 458).

7.27.15.5 Upgrading from a Previous Version

Migrating to version 1.3 from an earlier version involves automatic upgrade of the underlying database data, so it's a good idea to make database backup. If you decide to downgrade back to 1.2 after upgrade, you'll need to clear the database and restore from backup because 1.2 will likely not be able to work with the upgraded database.

1. Create Structure database backup(see page 486)
2. Upgrade the plugin from UPM or manually
3. When the plugin has been upgraded, verify the basic functionality works. You may need to fully refresh you browser page with Ctrl+Reload or Alt+Reload.
4. Open **Manage Structure** page and go over structures that have the following synchronizers installed:
    a. GreenHopper synchronizer: only important if you have upgraded to GreenHopper 5.8. Check if you see `Rank Field: ?` in the synchronizer configuration – this synchronizer should be deleted and installed again, because GreenHopper has changed the Ranking field.
    b. SubTasks synchronizer: make sure you're ok with the change in the synchronizer logic explained above (always forcing sub-tasks to be under their parent). If not, disable the synchronizer and run Resync manually when needed.

7.27.15.6 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 1.3[572].

7.27.15.7 Structure 1.3.1 Release Notes

> **10 November 2011**
>
> Structure 1.3.1 contains an important security fix.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[573]

1. Patch Release

This is a patch release based on version 1.3 with a fix for a high-severity security vulnerability. Additionally, one minor bug has been fixed.

571 https://plugins.atlassian.com/plugin/details/34717
572 http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.3%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true
573 https://plugins.atlassian.com/plugin/details/34717

1.1 Security Fix

A high-severity security vulnerability has been fixed in this release. The vulnerability is present in all previous versions of Structure. The details of the vulnerability, workarounds and the fix will be sent in a security advisory to the registered technical contacts. Security advisory will be made public two weeks after the release.

⚠ Commercial license owners: if you haven't received the security advisory, and if you are the technical contact for Structure in your company, please send an e-mail to support@almworks.com[574] and mention your JIRA Server ID or Structure license serial number.

1.2 Bug Fixed

We have fixed a bug[575] that prevented valid Structure license (evaluation or commercial) from working on some operating systems (such as Windows Server 2003) on some specific condition.

2. Supported JIRA Versions

This version supports JIRA versions 4.2 – 4.4.x. There are separate downloadable files for JIRA 4.4.x, 4.3.x and 4.2.x.

3. Upgrading

Upgrade to 1.3.1 is **highly recommended**. If your JIRA and Structure are available to random Internet users, the upgrade is critical.

Upgrade from version 1.3 is straightforward and does not require special actions. For upgrading from earlier versions, please read the related (HIDDEN) Release Notes (Structure 1-2)(see page 761).

⚠ Commercial license owners: if you have Structure versions 1.0 – 1.2, and do not wish or cannot upgrade to 1.3.1, please contact support[576]. Tell us your JIRA version, your Structure version and your JIRA Server ID – and we will create the security patch for your specific version of Structure.

7.27.15.8 Structure 1.3.2 Release Notes

ⓘ **29 November 2011**

Structure 1.3.2 is a patch release that contains a medium-severity bug/security fix.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[577]

574 mailto:support@almworks.com
575 http://jira.almworks.com/browse/HJ-512
576 mailto:support@almworks.com
577 https://plugins.atlassian.com/plugin/details/34717

1. Patch Release

This is a patch release based on version 1.3.1 with a fix for a medium-severity bug that also has security-related implications.

1.1 The Fix

A bug that presents a medium-severity security vulnerability has been fixed in this release. The vulnerability is present only in versions 1.3 — 1.3.1 of the Structure plugin. The details of the vulnerability and a workaround will be posted in a security advisory to the Structure Security Group[578]. The security advisory will be made public later.

2. Supported JIRA Versions

This version supports JIRA versions 4.2 – 4.4.x. There are separate downloadable files for JIRA 4.4.x, 4.3.x and 4.2.x.

3. Upgrading

Upgrade to 1.3.2 is recommended for everyone who has versions 1.3 or 1.3.1 installed. If your JIRA and Structure are available to random Internet users and you allow creating new structures to the general public, the upgrade is critical.

Upgrade from version 1.3 is straightforward and does not require special actions. For upgrading from earlier versions, please read the related (HIDDEN) Release Notes (Structure 1-2)(see page 761).

7.27.16 Structure 1.2 Release Notes

30 August 2011

Structure 1.2 introduces Structure Dashboard Gadget, Status Rollup Synchronizer and `structure()` JQL function.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[579]

7.27.16.1 Version Highlights

- **Structure Gadget** allows to add several structure widgets to JIRA dashboard and view structure in Confluence.
- **Status Rollup Synchronizer** changes status of a parent issue according to the statuses of its sub-issues. For example, you can make parent issue *Resolved* if all sub-issues are *Resolved*.
- `structure()` JQL function selects issues from structure and lets you build complex JQL queries that involve structures.

This version also contains a number of other minor improvements and bug fixes.

578 http://groups.google.com/group/structure-security-list
579 https://plugins.atlassian.com/plugin/details/34717

7.27.16.2 Changes in Detail

Structure Gadget

Structure Gadget lets you view and edit structure on a JIRA Dashboard and in Confluence.

You can see structure alongside other information, and you can also add several structures to the same dashboard or Confluence page.

The gadget displays the familiar structure component, albeit slightly stripped down — there are no secondary panels and no toolbar. Other than that, the structure widget is fully functional — unless you specifically make it read-only, you can change the structure and create and update new issues. (You'll need to use keyboard shortcuts without the toolbar.)

Documentation: Structure Gadget(see page 394), Using Structure Gadget in Confluence(see page 832)

Status Rollup Synchronizer

Status Rollup synchronizer allows you to make issue status change automatically when the statuses of its sub-issues change. The most popular example is to make the parent issue *Resolved* if all sub-issues are *Resolved*, but the synchronizer can do more than that.

The configuration lets you set up a **sequence of statuses** that the parent issue can be taken to. When analyzing the sub-issues, the synchronizer finds out the **earliest** status from this sequence, and tries to make the parent issue have this status. For example, if you have status sequence *(Open, Resolved, Closed)*, the synchronizer:

- makes parent issue *Open* if at least one sub-issue is *Open*,
- makes parent issue *Resolved* if all sub-issues are either *Resolved* or *Closed*, and
- makes parent issue *Closed* if all sub-issues are *Closed*.

The configuration also lets you limit the synchronizer's actions by picking applicable projects, issue types and which workflow transition may be used.

Documentation: Status Rollup Synchronizer(see page 534)

`structure()` JQL Function

Whenever you use JQL, you can now specify a condition **`issue in structure()`**, or **`issue in structure(structureName, parentIssue)`**, or other variation with `structure()` function. This function selects the issues from the given structure. If `parentIssue` parameter is specified, only that issue and its sub-issues (at any level) are selected.

Since you can combine structure condition with other conditions in a JQL query, this feature gives you more options for using Structure.

Documentation: S-JQL Reference(see page 272)

Type+Priority (TP) Column

Structure widget now offers a column named **TP**, which shows icons for issue type and priority while consuming minimal horizontal space.

Highlighting of Live Data Updates

Structure widget is live — it detects changes on the server and updates the issue grid. Now when a cell with some field value is updated, it is briefly highlighted with yellow.

Other Improvements and Notable Bugs Fixed

- Performance improved when polling and downloading data for the displayed issues.
- Bug fixed: HJ-433 Copy/Cut actions conflict with the browser's copy/cut, preventing copying of text[580]
- Bug fixed: HJ-431 Resync can be started by a user with Control permissions, but the status won't be shown[581]
- Bug fixed: HJ-430 Fixed view bug: issues appear between target issue and its parent[582]

7.27.16.3 Supported JIRA Versions

This version of Structure supports JIRA 4.2 — 4.4.x. Note that there are separate downloadable files for JIRA 4.4.x, 4.3.x and 4.2.x.

7.27.16.4 Installation

The plugin may be installed from the Plugin Exchange[583] or from the **Administration | Plugins** menu if you have JIRA 4.3 or Universal Plugin Manager. Further information is available in the Administrator's Guide(see page 458).

7.27.16.5 Upgrading from a Previous Version

Upgrading from versions 1.0-1.1.1 boils down to replacing the plugin JAR, either with "Upgrade" button in the Plugins administration section, or manually. All data and configuration is preserved. However, it's always a good idea to backup your Structure data(see page 486) or do a full JIRA backup **and** Structure backup.

7.27.16.6 Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 1.2[584].

580 http://jira.almworks.com/browse/HJ-433
581 http://jira.almworks.com/browse/HJ-431
582 http://jira.almworks.com/browse/HJ-430
583 https://plugins.atlassian.com/plugin/details/34717
584 http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.2%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true

7.27.17 Structure 1.1 Release Notes

29 July 2011

Structure 1.1 improves how structure looks on the issue page and adds more options for JIRA administrator.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[585]

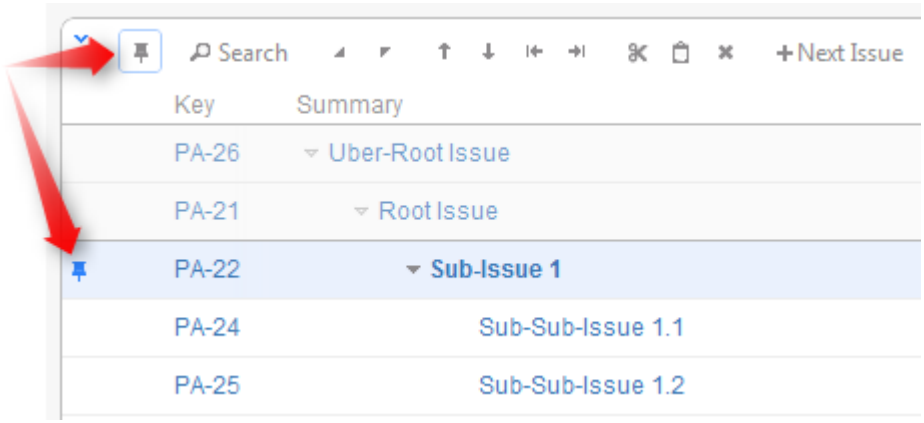
7.27.17.1 1. Version Highlights

- Structure Panel on the Issue Page by default displays only the viewed issue, its parents and sub-issues, but not other issues from the hierarchy.
- JIRA Administrator is now able to select which users can use Structure and create new structures.
- Multiple smaller improvements, bug fixes and performance optimizations.
- Preliminary version for JIRA 4.4.

7.27.17.2 2. Changes in Detail

2.1 Fixed Structure View

Pinned Item Mode(see page 192) allows the user to see only part of the structure that's relevant to a specific issue. Only the issue itself, its parent issues up to the top level and its sub-issues are displayed. Fixed Structure View can be turned on/off with the toolbar button or with **Ctrl+.** keyboard shortcut.



Documentation: Pinned Item Mode(see page 192).

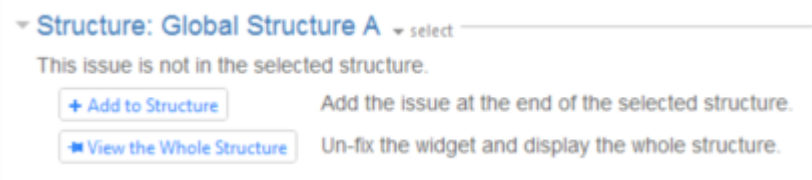
2.2 Better Structure on the Issue Page

Structure Panel on the Issue Page is now by default is fixed on the issue being viewed, using the new Fixed Structure View - displaying only part of the structure related to the viewed issue. This improves page loading time and makes the currently viewed issue stand out in the hierarchy.

---

585 https://plugins.atlassian.com/plugin/details/34717

Additionally, if the viewed issue is not present in the currently selected structure, the structure widget won't be displayed. You'll be able to quickly add the issue to the end of the structure, or choose to view the whole structure. You can also select another structure to view.

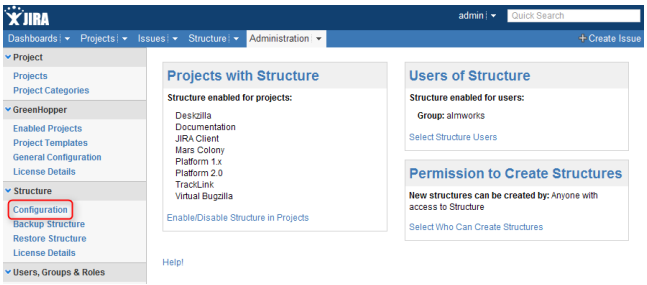


2.3 More Control for the Admin

JIRA Administrator is now able to control who has general access to the Structure plugin.

Firstly, you can restrict access to Structure by selecting which groups are allowed to use it. If a user does not belong to any of these groups, they will not see Structure elements in JIRA at all.

Secondly, you can restrict the ability to create new structures in the same way, by selecting user groups which are given that permission.



Documentation: Restricting User Access to Structure(see page 478), Changing Permission to Create New Structures(see page 479)

2.4 Loading Performance Improvement

We have made initial structure loading faster by improving performance of the permission checks. This is especially important for large structures, where thousands of issue IDs might need checking before they can be transferred to the user.

2.5 Help Links

Links to the online documentation are now scattered across various Structure pages, and **Help** link is also present in the structure widget's footer.

2.6 Notable Bugs Fixed

- Fixed: HJ-380 Structure panel and time tracking information are not present on the Issue Page if the issue's project has numbers in its Key[586]
- Fixed: HJ-363 Links synchronizer doesn't remove link from issue moved to top level[587]

586 http://jira.almworks.com/browse/HJ-380
587 http://jira.almworks.com/browse/HJ-363

7.27.17.3 3. Supported JIRA Versions

This version supports JIRA versions 4.1 – 4.3.x. There are separate downloadable files for JIRA 4.3.x, 4.2.x and 4.1.x.

Preliminary version for JIRA 4.4 is available in Download(see page 978) section.

> ⚠ Support for JIRA 4.1 will be dropped in the first Structure release that follows the release of JIRA 4.4.

7.27.17.4 4. Installation

The plugin may be installed from the Plugin Exchange[588] or from the **Administration | Plugins** menu if you have JIRA 4.3 or Universal Plugin Manager. Further information is available in the Administrator's Guide(see page 458).

7.27.17.5 5. Upgrading from a Previous Version

Upgrading from version 1.0 boils down to replacing the plugin JAR, either with "Upgrade" button in the Plugins administration section, or manually. All data and configuration is preserved. However, it's always a good idea to backup your Structure data(see page 486) or do a full JIRA backup **and** Structure backup.

Upgrading from an earlier version should also run smoothly, but it will likely involve data migration, so backup is strongly advised.

7.27.17.6 6. Detailed Change List

Our public JIRA contains the full list of issues resolved in Structure version 1.1[589].

7.27.17.7 Structure 1.1.1 Release Notes

> ⓘ **8 August 2011**
>
> Structure 1.1.1 is a bug-fix release with a few non-critical bugs fixed since version 1.1 and with full support for JIRA 4.4.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[590]

1. Patch Release

This is a patch release with several bugfixes based on version 1.1 - please read Structure 1.1 Release Notes(see page 836) if you are upgrading from an older version.

---

588 https://plugins.atlassian.com/plugin/details/34717
589 http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.1%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true
590 https://plugins.atlassian.com/plugin/details/34717

1.1 Notable Bugs Fixed

- Fixed: HJ-386 Wiki-rendered images and other block / replaced elements in description ruin widget layout[591]

Our public JIRA contains the full list of issues resolved in Structure version 1.1.1[592].

2. Supported JIRA Versions

This version supports JIRA versions 4.2 – 4.4.x. There are separate downloadable files for JIRA 4.4.x, 4.3.x and 4.2.x.

Support for JIRA 4.1 is dropped starting with this version. The last working version for JIRA 4.1 is Structure 1.1. You can download older versions of Structure from Download Archive(see page 979).

3. Upgrading

Upgrading from version 1.1 should be straightforward, without the need for any extra preparation. If upgrading from an older version, please read Structure 1.1 Release Notes(see page 836) first.

7.27.18 Structure 1.0 Release Notes

13 July, 2011

Structure 1.0 release is the first stable release of the Structure Plugin, bringing production-quality hierarchical lists of issues to Atlassian JIRA.

Download the Latest Version(see page 978)
Structure on Plugin Exchange[593]

7.27.18.1 1. Product Highlights

- Multiple, shareable **hierarchical lists**
- Hierarchy of **any depth** (independently from subtasks and subtask hierarchy)
- Hierarchy can span **multiple projects** and not limited to specific issue types
- **Time tracking aggregation** on the issue page
- Real-time **collaboration** with live data updates
- Synchronization with **GreenHopper**, **Issue Links**, **Sub-tasks** and **Saved Filters**
- Powerful user interface with **keyboard shortcuts**

7.27.18.2 2. Changes from Structure 1.0-rc1

We have kept changes to a minimum to keep this release stable, adding only bugfixes and high-priority or low-impact improvements.

591 http://jira.almworks.com/browse/HJ-386
592 http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.1.1%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true
593 https://plugins.atlassian.com/plugin/details/34717

2.1 Documentation

A full set of documentation has been created and now available at http://wiki.almworks.com/display/structure

2.2 Improvements

- GreenHopper synchronizer(see page 839) allows to select non-global Rank and Epic/Theme fields (enabled for the selected project.)
- Progress column(see page 300) now shows numerical % value in the tooltip if you keep the mouse pointer over it.
- *Description* column is added to the list of available columns, which lets you print description from the Structure's printable page(see page 380).
- Bundled synchronizers (GreenHopper and Links) now write synchronization undo files(see page 547) to offset the risk of massive unintended changes due to incorrect synchronizer configuration. The undo files may be processed by a script to reverse the changes to issue data.

2.3 Notable Bugs Fixed

- Server exception when many projects are enabled for Structure in the administration interface
- Missing expand/collapse icons in the Internet Explorer 7
- GreenHopper Synchronizer with "Force Sub-tasks" mode allows placing sub-tasks at incorrect positions in the structure
- Text search does not find fully spelled words when indexing language is other than "other"

7.27.18.3 3. Supported JIRA Versions

Structure 1.0 is supported on JIRA versions 4.1 – 4.3.x. There are separate versions of the plugin for JIRA 4.3.x, 4.2.x and 4.1.x.

A version with preliminary support for JIRA 4.4 Release Candidate is available upon request.

Support for JIRA 4.1 will be dropped in the first Structure release that follows the release of JIRA 4.4.

7.27.18.4 4. Installation and Upgrade

The plugin may be installed from the Plugin Exchange[594] or from the **Administration | Plugins** menu if you have JIRA 4.3 or Universal Plugin Manager. Further information is available in the Administrator's Guide(see page 458).

Upgrading from version 1.0-rc1 boils down to replacing the plugin JAR, either with "Upgrade" button in the Plugins administration section, or manually. All data and configuration is preserved. However, it's always a good idea to backup your Structure data(see page 486) or do a full JIRA backup.

Upgrading from earlier versions should also run smoothly, but it will likely involve data migration, so backup is strongly advised.

---

594 https://plugins.atlassian.com/plugin/details/34717

7.27.18.5 5. Detailed Change List

The list of issues resolved with this release is available in our JIRA[595].

[9] http://jira.almworks.com/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%3D+HJ+AND+issuetype+in+%28Bug%2C+Improvement%2C+%22New+Feature%22%29+AND+fixVersion+%3D+%221.0%22+AND+status+in+%28Resolved%2C+Closed%29+ORDER+BY+votes+DESC%2C+issuetype+DESC%2C+priority+DESC&runQuery=true&clear=true&reset=true

8 Structure Developer's Guide

8.1 Structure Developer Documentation

- Structure Concepts, Developer's Perspective(see page 843)
- Accessing Structure from JIRA Plugin(see page 845)
    - Setting Up the Integration(see page 845)
        - Additional Libraries Used in Structure API(see page 846)
        - Controlling Compatibility(see page 847)
        - Making Structure Dependency Optional(see page 848)
    - Structure Services(see page 849)
    - Building Forest Specification(see page 851)
    - Reading Structure Content(see page 853)
    - Changing Structure Content(see page 855)
    - Loading Attribute Values(see page 858)
    - Creating and Adding Folders(see page 860)
    - Creating Dynamic Structures(see page 861)
- Extending Structure Functionality(see page 861)
    - Creating a New Column Type(see page 862)
    - Loading Additional Web Resources For Structure Widget(see page 879)
    - Declaring a New Generic Item Type(see page 880)
- Accessing Structure Data Remotely(see page 882)
- Reference(see page 882)
    - Structure Java API Reference(see page 883)
        - Structure API Versions(see page 883)
    - Structure Plugin Module Types(see page 888)
        - structure-attribute-loader-provider(see page 889)
        - structure-export-renderer-provider(see page 889)
        - structure-item-type(see page 891)
        - Generator Modules(see page 891)
        - structure-effector(see page 893)
        - structure-effect-provider(see page 894)
        - structure-query-constraint(see page 895)
        - new-structure-template(see page 895)
    - Structure REST API Reference(see page 896)
        - Structure Resource(see page 897)
        - Forest Resource(see page 917)
        - Item Resource(see page 920)
        - Value Resource(see page 924)
        - Attribute Subscription Resource(see page 928)
        - Configuration Resource(see page 933)
    - Structure JavaScript API Reference(see page 944)
        - JavaScript API Functions(see page 944)
        - JavaScript API Classes(see page 946)
            - Column Class(see page 947)
            - ColumnConfigurator Class(see page 951)
            - ColumnOption Class(see page 953)
            - ColumnType Class(see page 956)
            - ItemDetailsProvider Class(see page 958)
        - JavaScript API Objects(see page 962)

        - ItemDetailsBridge Object(see page 962)
    - Web Resource Contexts(see page 965)
- API Usage Samples(see page 965)
- Archived Documentation(see page 966)
    - Structure 3 API Changes(see page 967)
    - Creating a New Synchronizer(see page 973)
    - structure-synchronizer(see page 974)
- Open Source Licenses(see page 975)

Structure for Developers

Structure app provides APIs that allow you to access structures, integrate your add-on with Structure and extend Structure functionality. Here are the typical use cases:

Custom Development

You customize JIRA for your customer or employer, and you need to integrate Structure with some other in-house system – see section about integrating plugins(see page 845) and Java API reference(see page 883).

Plugin Integration

You have your own great JIRA app, or plan to create one, and you'd like to use the issue hierarchy provided by Structure – see Accessing Structure from JIRA Plugin(see page 845).

Extending Structure

You'd like to extend Structure, adding functionality to the plugin itself – read documentation about extending Structure functionality with additional plugins(see page 861).

Remote Access

You need to get or change issue hierarchy remotely from some automated scripts or a client application – read about Accessing Structure Data Remotely(see page 882) and Structure REST API(see page 896).

8.2 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.

8.2.1 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.

8.2.2 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.

8.3 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.

- Setting Up the Integration(see page 845)
- Structure Services(see page 849)
- Building Forest Specification(see page 851)
- Reading Structure Content(see page 853)
- Changing Structure Content(see page 855)
- Loading Attribute Values(see page 858)
- Creating and Adding Folders(see page 860)
- Creating Dynamic Structures(see page 861)

8.3.1 Setting Up the Integration

To start using Structure in your plugin:

8.3.1.1 Add dependency to your `pom.xml`

Figure out the version of the API(see page 883) that you need – it may depend on your JIRA and Structure plugin version.

To use API classes, add the following dependency:

```
<dependency>
  <groupId>com.almworks.jira.structure</groupId>
  <artifactId>structure-api</artifactId>
  <version>17.0.0</version>
  <scope>provided</scope>
</dependency>
```

Note that there are Additional Libraries Used in Structure API(see page 846)

8.3.1.2 Import `StructureComponents`

In your `atlassian-plugin.xml`, use `<component-import>` module to import `StructureComponents` service. This service provides access to all other Structure services.

Alternatively, you can import specific services.

```
<component-import key="structure-components" interface="com.almworks.jira.structure.api.StructureComponents"/>
```

8.3.1.3 Have Structure API service injected into your component

```
public class MyClass {
  private final StructureManager structureManager;

  public MyClass(StructureComponents structureComponents) {
    structureManager = structureComponents.getStructureManager();
  }

  ...
}
```

This is it! Continue to the list of Structure Services(see page 849) to see which service you need to work with. Other articles in this section provide examples for specific use cases.

For a production plugin, consider Controlling Compatibility(see page 847). For a standalone plugin, which can work without Structure, read about Making Structure Dependency Optional(see page 848).

8.3.1.4 Additional Libraries Used in Structure API

Structure API has dependencies on a few open-source libraries that are transitively included in your project when you add a dependency on Structure API.

You don't need to explicitly add dependencies on these libraries.

Integers and HPPC

The open source library Integers[596] 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[597], which provides specific implementations of these collections.

See API Usage Samples(see page 965) 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.

596 http://code.google.com/p/integers/
597 http://labs.carrotsearch.com/hppc.html

8.3.1.5 Controlling Compatibility

Why Declare Compatible Versions

Structure Java API will change with time, and it is a good practice to ensure that your plugin uses the correct version of the API.

Structure API Versions(see page 883) page explains how version numbers change based on how compatibility is affected. Say, you develop your code using Structure API version 16.2.0 – your code will work with any version of the API starting from 16.2.0 and up to, but not including version 17.0.0.

So what happens if your code is run on JIRA with Structure that provides an incompatible API? It may break, or it may work. The exact answer depends on which parts of the API you use and what are the differences. But if the code breaks, it may not break outright – it may seem to work at first, until it tries to use a method that's not there, for example.

To make your code fail fast, you can declare dependency on a specific range of versions of the Structure API. In that case, if the version of the API is different, your plugin will fail to load and the user will immediately know that there's a problem.

Importing Specific Range of API Versions

You can declare dependency on the specific range of the API versions via OSGi bundle instructions added to your `pom.xml` or `atlassian-plugin.xml`. Figure out the compatible OSGi versions range from the API versions(see page 883) table and modify your `pom.xml` to contain the following:

```
<plugin>
  <groupId>com.atlassian.maven.plugins</groupId>
  <artifactId>maven-jira-plugin</artifactId>
  ...
  <configuration>
    <instructions>
      <Import-Package>
        com.almworks.jira.structure.api*;version="[16,17)",
        com.almworks.integers*;version="0",
        org.jetbrains.annotations;version="0"
      </Import-Package>
    </instructions>
  </configuration>
</plugin>
```

Here we are declaring the acceptable range of versions for the Structure classes, taken from the example above. We don't much care about the versions of Integers and Annotations libraries, so `version="0"` will match any version of those packages.

You may have other `Import-Package` instructions to declare dependency rules for other packages, and you may have other instructions besides `Import-Package` as well. See the API Usage Samples(see page 965) for a more complete example.

Next: Making Structure Dependency Optional(see page 848)

8.3.1.6 Making Structure Dependency Optional

If you are integrating your plugin with Structure, or when you generally write code that uses Structure API but also should work when Structure Plugin is not present, you need to declare that dependencies are optional and isolate dependencies in the code.

Declare Optional Dependency

Since your plugin must first be loaded as an OSGi bundle, it should declare dependencies from the Structure API packages as optional.

Modify `<Import-Package>` declaration in your `pom.xml` or `atlassian-plugin.xml` and add `resoltion:=optional` classifier. (Add Import-Package to control API compatibility(see page 847) if you don't have this declaration yet.)

```
<Import-Package>
  com.almworks.jira.structure*;version="[16,17)";resolution:=optional,
  com.almworks.integers*;version="0";resolution:=optional,
  org.jetbrains.annotations;version="0";resolution:=optional
</Import-Package>
```

Isolate Dependencies in the Code

So once you have declared the optional resolution of the Structure API classes, your bundle will load - but if your code tries to access a class from the Structure API, you'll get a `NoClassDefFoundError`. To avoid that, you need to isolate the dependency on Structure API classes - typically in some wrapper classes.

This is also a point to make design decisions. So your code can use Structure when it's present, and can work independently when Structure is not there. Are there any abstractions that address both of these situation? What are the concepts that are realized through Structure API and through some other means when Structure is not avialable?

Here's a sample wrapper for the Structure API that provides `ForestAccessor` wrapper (whatever it does) when Structure is available and `null` otherwise.

```
public class StructureAccessor {
  public static boolean isStructurePresent() {
    if (!
ComponentAccessor.getPluginAccessor().isPluginEnabled("com.almworks.jira.structure"))
{
      return false;
    }
    try {
      Class.forName("com.almworks.jira.structure.api.StructureComponents");
    } catch (Exception e) {
      return false;
    }
    return true;
  }

  public static ForestAccessor getForest(long structureId) {
    if (!isStructurePresent()) return null;
    StructureComponents structureComponents;
    try {
      structureComponents =
ComponentAccessor.getOSGiComponentInstanceOfType(StructureComponents.class);
    } catch (Exception e) {
      return null;
    }


    try {
      return new

ForestAccessor(structureComponents.getForestService().getForestSource(ForestSpec.stru
cture(structureId)));
    } catch (StructureException e) {
      return null;
    }
  }
}
```

8.3.2 Structure Services

This page lists public services provided by Structure API. All these services are available from StructureComponents[598] instance.

598 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/StructureComponents.html

8.3.2.1 Services to Start With

Use ...	to ...
StructureManager[599]	Create and delete structures, modify structure properties such as name or permissions. (But not to work with the structure's content.)
ForestService[600]	Access forests for reading or changing.
StructureAttributeService[601]	Retrieve attribute values for given rows in a given forest.
RowManager[602]	Extract item information for rows read from a Forest.
FolderManager[603]	Create folders or change folder properties.
GeneratorManager[604]	Create generators or change generator properties.

8.3.2.2 More Power

Use ...	to ...
StructureConfiguration[605]	Change global Structure add-on configuration.
StructureViewManager[606]	Create and manipulate views.
StructureSyncManager[607]	Manage synchronizers.
StructureBackupManager[608]	Backup complete Structure data to a file or restore it back.
StructureFavoriteManager[609]	Read or change which structures are favorite of which users.

599 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/structure/StructureManager.html
600 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestService.html
601 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/StructureAttributeService.html
602 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/row/RowManager.html
603 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/folder/FolderManager.html
604 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/generator/GeneratorManager.html
605 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/settings/StructureConfiguration.html
606 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/view/StructureViewManager.html
607 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/StructureSyncManager.html
608 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/backup/StructureBackupManager.html
609 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/structure/favorite/StructureFavoriteManager.html

Use ...	to ...
PropertyService[610]	Store arbitrary properties.
StructurePropertyService[611]	Store arbitrary per-structure properties.
AttributeSubscriptionService[612]	Create a subscription for a set of attributes and rows and load data for them asynchronously.

8.3.2.3 Extreme Power

Use ...	to ...
ItemTracker	Track recorded changes that happened to items (in JIRA Data Center – on all nodes of the cluster).
ItemResolver	Convert `ItemIdentity` into an object representing that item.
IssueEventBridge	Listen for or report issue events.
StructureQueryParser	Parse an S-JQL query.
StructureQueryBuilderFactory	Build an S-JQL query via Builder pattern.
ProcessHandleManager	Manage feedback page for asynchronous processes.
SyncAuditLog	Access or manage Synchronization Audit log.
StructureJobManager	Run a job asynchronously.
ScheduledJobManager	Schedule a periodical job to run asynchronously (only on a single node in a cluster).

8.3.3 Building Forest Specification

A forest specification, or `ForestSpec`[613], is a way for your code to identify the forest that you'd like to access. The forest may come from different sources – it could be a structure, it could be a transformed(see page 185) structure, it could be a result of query or some other types of forest source.

610 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/property/PropertyService.html
611 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/property/StructurePropertyService.html
612 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/subscription/AttributeSubscriptionService.html
613 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestSpec.html

So the first step before you read or update a forest is to create an instance of ForestSpec. Here are some examples of how you can do that.

Desired forest	ForestSpec expression
Base Content	
Structure #123	`ForestSpec.structure(123)`
Result of a JQL query	`ForestSpec.sQuery("jql", "priority = Blocker")`
Result of a text query	`ForestSpec.sQuery("text", "text to find")`
Adjusted Content	
Structure #123, sorted by Priority	`ForestSpec.structure(123).transform( CoreStructureGenerators.SORTER_ATTRIBUTE, ImmutableMap.of( "attribute", (Object) ImmutableMap.of("id", IssueFieldConstants.PRIORITY, "format", "order") "desc", true ) );`
Structure #123, skeleton only (without dynamic content)	`ForestSpec.skeleton(123)`
Structure #123, with title row	`ForestSpec.skeleton(123).withTitle()`

More details are available in Javadocs for ForestSpec[614].

614 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestSpec.html

8.3.4 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.

8.3.4.1 Figure out Structure ID

To address a structure, you need to know its ID. If you just have a name you can do the following:

```
List<Structure> structures =
myStructureComponents.getStructureManager().getStructuresByName("My Structure",
PermissionLevel.VIEW);
long structureId;
if (structures.size() == 1) {
  structureId = structures.get(0).getId();
} else {
  // no structures or too many structures -- error?
}
```

Now you have `structureId` or an error situation where the name does not uniquely identify your structure.

8.3.4.2 Create a ForestSpec

You need a forest specification to get a ForestSource. You can read more about this in the section about Building Forest Specification(see page 851).

```
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.

8.3.4.3 Retrieve ForestSource

A `ForestSource` is an interface that produces some specific forest and that provides versioning for it.

```
ForestSource forestSource =
myStructureComponents.getForestService().getForestSource(forestSpec);
```

Note that this call may produce `StructureException` in case a structure cannot be found and in some other cases. A robust code would have some exception handling.

⚠ Do not store a `ForestSource` in memory for a long time, longer than a single user request. Structure has internal caching engine that efficiently manages forest sources and their dependencies. Request forest source from `ForestService` in every new request.

8.3.4.4 Retrieve Forest and its version

Forest source can provide you with the latest version of the forest, or with an incremental update, based on the version you already have.

To get the latest forest:

```
VersionedForest versionedForest = forestSource.getLatest();
DataVersion latestVersion = versionedForest.getVersion();
Forest forest = versionedForest.getForest();
```

Note that `latestVersion` variable contains the version of the forest that you got. You can later use it to call `forestSource.getUpdate(latestVersion)` and receive only information about how did the forest change since the last time you've seen it.

⚠ You cannot really use `latestVersion` for anything else besides getting updates later. The numbers in that version bear no meaning regarding structure's history. For history queries, you'll need to use `HistoryService`.

8.3.4.5 Iterate through Forest and get StructureRow instances

A `Forest` is just two parallel arrays, one containing row IDs, the other containing depths. (Or, one can say that it is a list of pairs `(rowId, depth)`.) You can iterate through it via simple cycle.

For each row, you'll need more information than just row ID. We use `RowManager` to retrieve other properties of a row.

```
RowManager rowManager = myStructureComponents.getRowManager();
for (int i = 0; i < forest.size(); i++) {
  long rowId = forest.getRow(i);
  int depth = forest.getDepth(i);
  StructureRow row = rowManager.getRow(rowId);
  ...
}
```

Note that `row` is never `null`, because Row Manager would through an unchecked exception if a row is not found – this situation is considered a developer's error.

8.3.4.6 Analyze the row and process data

Finally, you get `ItemIdentity` from the row to understand which item does the row show. The items could be anything – issues, folders, users. So even if your structure only contains issues, it is advised to do an extra check.

```
ItemIdentity itemId = row.getItemId();
if (CoreIdentities.isIssue(itemId)) {
  long issueId = itemId.getLongId();
  // process the row!
  ...
}
```

A structure with dynamic content will also contain generators. If you take all the rows, regardless of the item type and use them somewhere, you might stumble upon a generator. To eliminate them from the analyzed forest, add a condition. The same is usually done for "loop markers", which are special items added by extenders to indicate that there's a loop (like cyclic issue links).

```
  ItemIdentity itemId = row.getItemId();
  if (!CoreIdentities.isGenerator(itemId) && !
CoreIdentities.isLoopMarker(itemId)) {
    ...
  }
```

Congratulations! You've successfully implemented forest read-out.

You can adjust this walkthrough for your needs – for example, read a query result, or read only a portion of a forest.

8.3.5 Changing Structure Content

Updating a structure can be done through the same `ForestSource` interface that was used for Reading Structure Content(see page 853). In this article, we're assuming that you've got `forestSource` local variable that you've created according to instructions in the previous article.

8.3.5.1 Forest Coordinates

To make a change to a forest, you need to be able to point to a specific part of a forest. This is done by using row IDs, which uniquely identify forest rows.

- To point to a specific row in the forest, which you'd like to move or delete, you just use this row's ID.
- To point to a specific position in the forest, where you'd like to insert or move rows to, you need to use row IDs of its neighbors, or *coordinates*:
  - "Under" coordinate is the row ID of the future parent of the inserted row, or zero if the row is placed at the top level.
  - "After" coordinate is the row ID of the future preceding sibling of the inserted row under the same parent, or zero if the row is placed as the first child.
  - "Before" coordinate is the row ID of the future succeeding sibling of the inserted row under the same parent, or zero if the row is placed as the last child.

8.3.5.2 Applying Forest Action

To make a change, you need to call `forestSource.apply()` method, passing a specific `ForestAction` that you want to apply.

Adding a single row

To add a single row to the forest, use `ForestAction.Add` constructed with the `ItemIdentity` of the item associated with that row.

```
forestSource.apply(new ForestAction.Add(CoreIdentities.issue(10000), under, after,
before))
```

Adding a sub-forest

To add multiple rows in one action, use `ForestAction.Add` that receives an `ItemForest`.

`ItemForest` is a special container that is used to build a temporary forest with temporary rows, having negative row IDs. The class provides information both about the hierarchy of inserted temporary rows (via `Forest`) and a mapping from the temporary row ID to the inserted `ItemIdentity`.

To create an `ItemForest`, you need to use either `ImmutableItemForest` or `ItemForestBuilderImpl`.

```
ItemForest itemForest = new ItemForestBuilderImpl()
  .nextRow(CoreIdentities.textFolder("My Issues"))
  .nextLevel()
    .nextRow(CoreIdentities.issue(10000))
    .nextRow(CoreIdentities.issue(10001))
    .build();
forestSource.apply(new ForestAction.Add(itemForest, under, after, before));
```

Removing a sub-tree

To remove a row, use `ForestAction.Remove` and pass the row ID being removed.

```
forestSource.apply(new ForestAction.Remove(LongArray.create(100, 101, 102)));
```

⚠ All sub-rows of the removed rows will be removed as well. If you need to keep them, apply `ForestMove` on them first.

Moving a sub-tree

To move a row with its sub-rows, use `ForestAction.Move`.

You can specify one or more row IDs, which can be from the different parts of the forest. Those rows will be placed one after another at the specified position.

```
forestSource.apply(new ForestAction.Move(LongArray.create(100, 101, 102), under,
after, before));
```

8.3.5.3 Inspecting the Results

A call to `ForestSource.apply()` will finish successfully if the operation has been completed and throw a `StructureException` otherwise.

You can inspect the returned `ActionResult` to get information about the *effects* of the action (more on effects below).

You can also use `ActionResult.getRowIdReplacements()` – it is a mapping from the temporary row IDs, used when adding rows, to the newly assigned real row IDs, which are now part of the structure.

8.3.5.4 Effects and Changing Dynamic Structures

You may have noticed that you can apply actions to any forest source, not necessarily a simple structure. It can be a transformed structure, or even a transformed query. A structure can also contain dynamic parts, created or adjusted by generators, and you can try to apply the actions that would affect these parts.

A successful action would produce one or more *Effects* (represented in the `ActionResult` as `AppliedEffect`). In simple case of changing a non-dynamic structure, it would be, unsurprisingly, a structure change. In case the action involves dynamic content, the effects may differ – but the general concept is that, after the effect takes place, the updated (re-generated) structure will reflect the desired action's result.

Here are some examples of the possible effects.

Action	Effect
Adding rows to a static structure	✅ Structure is modified
Moving item X from group A to group B, where groups are provided by a grouper by field F	✅ The value of F for X is changed from A to B
Removing issue X from under issue Y, when previously X was added automatically by a Links Extender using link type L	✅ Link L: Y → X is deleted
Moving issue upwards when structure is sorted by Agile Rank	✅ Issue's Rank is changed
Adding an issue to an arbitrary JQL query result	⛔ StructureException is thrown – no way to force an issue to be part of a JQL result
Adding issue X under issue Y within the scope of a Links Extender and when issue Y is "static" (not added by the extender)	⛔ StructureInteractionException is thrown – there are two ways to interpret this action

As generators are extensible and can be added by other plugins, the range of possible effects is not limited.

Note that in the last two examples the action is not successful. In the last example, you need to use `ForestSource.apply()` with parameters, which would define whether a generator should process the action or if the issue should be inserted into the static structure.

8.3.5.5 Concurrency and Atomicity

Each `ForestAction` can be viewed as a separate transaction. It is atomic, meaning that it is either fully successful or fully failed.

There's no way to make a transaction larger. In other words, if you apply two actions to a forest source, it is possible that a concurrent action, done from another thread, is executed in between your two actions.

8.3.5.6 Permissions

All actions are executed under the "current" user and with all necessary permission checks. Updating a structure requires `EDIT` permission on the structure. Other effects, like changing issue fields, would require `EDIT_ISSUE` permission on the subject issues.

When permissions are insufficient, the action will not succeed and a `StructureException` will be thrown.

> When it comes to effects applied by generators, it is a generator's responsibility to check permissions before applying an action. All generators bundled with Structure have strict permission checks.

The current user is generally managed by JIRA and is the same as the user who makes the request. However, you can use `StructureAuth` class to "sudo" to another user or to bypass permission checks altogether.

8.3.6 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`.

8.3.6.1 About Attributes

One of the core concepts in Structure is the Attribute abstraction. An attribute is something that can provide a value of specific type and meaning for any row in a forest.

For example, a "Summary" attribute would produce the value of Summary field for issues, the name of a folder for folders and a person's full name for users. Some attributes may be applicable only to certain item types and would provide empty value for all other items.

Besides item-based attributes, which provide values that depend only on the item in the forest, there are forest-based attributes, which are calculated based on the whole forest and items in it.

> Forests and Attributes are two main concepts that make up the Structure grid. Looking at the Structure Board, you see Forest in the vertical direction – rows and hierarchy are taken from Forest, and you see Attributes in the horizontal direction – all columns load Attributes from the server and display those values.

8.3.6.2 General Approach to Loading Values

Let's assume that, after Reading Structure Content(see page 853), you have `StructureComponents` instance and an instance of `ForestSpec` for a forest. We can read a number of attributes for a number of rows by going to `StructureAttributeService`.

Figure out which Attributes do you need

The service accepts multiple attribute specs in one request. If you need several attributes calculated – it's better to do that in one request.

```
List<AttributeSpec<?>> attributeSet = new ArrayList<>();
attributeSet.add(CoreAttributeSpecs.KEY);
attributeSet.add(CoreAttributeSpecs.SUMMARY);
attributeSet.add(CoreAttributeSpecs.TOTAL_REMAINING_ESTIMATE);
```

`CoreAttributeSpecs` class and its parent class, `SharedAttributeSpecs`, contain some of the most popular attributes.

It's likely that you'll need to build you own attribute specification. For example, to address a numeric JIRA custom field and calculate total of that field based on sub-issues, you'll need the following.

```
AttributeSpec<Number> customField =
  AttributeSpecBuilder.create("customfield",
ValueFormat.NUMBER).params().set("fieldId", 10000).build();

AttributeSpec<Number> customFieldTotal =
  AttributeSpecBuilder.create(CoreAttributeSpecs.Id.SUM,
ValueFormat.NUMBER).params().setAttribute(customField).build();

attributeSet.add(customFieldTotal);
```

Figure out which Rows do you need to calculate the Attributes for

For example, this could be all rows in that structure.

```
LongList rows =
myStructureComponents.getForestService().getForestSource(forestSpec).getLatest().getF
orest().getRows();
```

If you need to create a `LongList` manually, use `LongArray` implementation.

Call `StructureAttributeService`

This service calculates a matrix of values for each row and attribute you specify.

```
RowValues values =
myStructureComponents.getAttributeService().getAttributeValues(forestSpec, rows,
attributeSet);
```

There is a variation of `getAttributeValues()` method that accepts a `Forest`, rather than `ForestSpec`. It is recommended to use the variant that accepts `ForestSpec` whenever possible, because that variant uses caching.

Read out the result

The returned object contains values for all pairs of requested row and requested attribute.

```
for (LongIterator ii : rows) {
  String key = values.get(ii.value(), CoreAttributeSpecs.KEY);
  Number total = values.get(ii.value(), customFieldTotal);
  ...
}
```

8.3.7 Creating and Adding Folders

You may need to create a new folder and add it to a structure.

Folders and generators are items that are managed entirely by Structure add-on, so you'll need to use Structure's services to create the item first, giving you the item identify, and then insert a row into a forest.

Read more about Changing Structure Content(see page 855) for general ideas about updating a structure.

8.3.7.1 Create the Folder entity

```
long folderId = myStructureComponents.getFolderManager().createFolder(Folder.named("M
y Stuff").build());
```

The folder is now stored in the database.

8.3.7.2 Define folder's identity

```
ItemIdentity itemId = CoreIdentities.folder(folderId);
```

8.3.7.3 Add folder to structure

```
forestSource.apply(new ForestAction.Add(itemId, 0, 0, 0));
```

8.3.8 Creating Dynamic Structures

Structures may have dynamic content, produced by generators.

Generators can be added to structure and moved around in the same way other items are added, as described in Changing Structure Content(see page 855). 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.

8.3.8.1 Create generator instance

You create a generator instance by calling `GeneratorManager`.

```
long generatorId = myStructureComponents.getGeneratorManager().createGenerator(
  CoreStructureGenerators.SORTER_AGILE_RANK,
  ImmutableMap.of(CoreGeneratorParameters.SORT_DESCENDING, false),
  structureId);
```

Note the third parameter – the generator is "owned" by a structure, so we should pass the ID of the owning structure.

8.3.8.2 Insert generator into the forest

Find parent row under which you'd like the forest to be automated. To apply generator to the whole forest insert generator at the top level by making "under" coordinate zero.

Do not use "after" and "before" coordinates unless you are adding an Inserter.

```
forestSource.apply(new ForestAction.Add(CoreIdentities.generator(generatorId), under,
0, 0));
```

This is it! Next time you read the contents of this forest source, it will have the results of this generator applied.

8.4 Extending Structure Functionality

You can extend Structure add-on's functionality with your own add-on by using one of the available extension points.

- Creating a New Column Type(see page 862)
- Loading Additional Web Resources For Structure Widget(see page 879)
- Declaring a New Generic Item Type(see page 880)

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[615] and we'll provide advice.

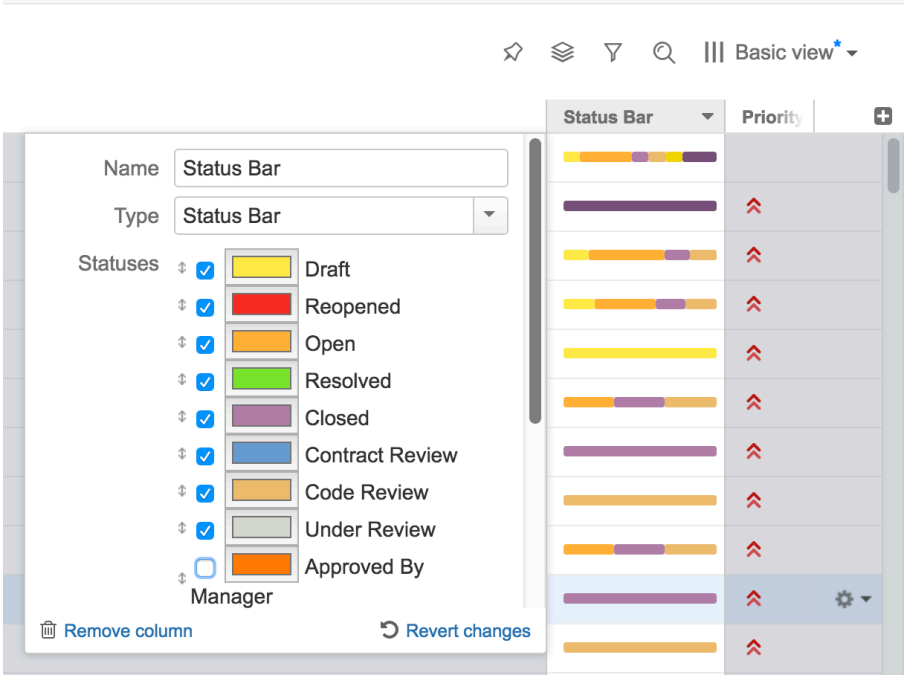
8.4.1 Creating a New Column Type

In this tutorial we will develop the Status Bar column type, which shows a progress-like bar filled with color stripes, each stripe's color representing a particular issue status, and each stripe's width being proportional to the number of issues having that status in the current issue's subtree.

You can download both the compiled plugin and its source code from API Usage Samples(see page 965).

- The Plan(see page 863)
- The Attributes(see page 863)
- AttributeSpec for Status Bar(see page 864)
- Status Bar Attribute(see page 864)
- Attribute Provider(see page 866)
- Client-Side Column(see page 867)
  - API Overview(see page 867)
  - Column Specifications(see page 868)
  - The Column Context(see page 869)
  - Requesting and Using Metadata(see page 871)
  - Column(see page 872)
  - ColumnConfigurator(see page 873)
  - ColumnOption(see page 873)
  - ColumnType(see page 875)
  - Column Groups(see page 876)
  - Web Resources and Contexts(see page 876)
- Export Renderers(see page 876)
  - Export Strategies(see page 877)
  - Generic Renderer Provider(see page 878)
  - Advanced Excel Renderer Provider(see page 878)

615 mailto:support@almworks.com



8.4.1.1 The Plan

A column type consists of several components. The client-side components are written in JavaScript and have two responsibilities:

- Rendering the cells in the Structure widget.
- Providing the column configuration UI.

The server-side components are written in Java and responsible for:

- Providing the attributes needed by the client-side part to render the cells.
- Exporting the column into printable HTML and Microsoft Excel formats.

For the Status Bar column we'll need to write code to cover all of the above responsibilities.

In general, however, only the client-side part is strictly necessary. If the attributes provided by Structure(see page 0) are enough for your column, you can skip the server-side attribute provider. You can also skip the components related to export, if this functionality is not critical. In that case, you can jump straight to the client-side part(see page 867), consulting the other chapters as necessary. For the complete treatment, please continue reading from top to bottom.

8.4.1.2 The Attributes

Before we begin, let's decide which attributes we need to pass from the server side to render a status bar. Obviously, the status bar depends on the statuses of all the issues in the given issue's subtree. This suggests that we need to use an "aggregate" attribute, and because Structure does not provide such an aggregate out of the box, we'll need to write our own.

Secondly, the colors and the order of statuses in the status bar are only a presentational matter. If we had a map from status IDs to sub-issue counts in the given issue's subtree, we could count the total number of sub-issues, scale the colored stripes so that they'd fill the whole status bar, and render them in any given order.

Thirdly, the "Include itself" option is somewhat trickier. When it's on, the current issue's status is shown in its status bar, as if there is one more sub-issue. When it's off, the current issue is excluded, and the status bar shows only its sub-issues (on all levels). We could try to implement this on the server side as a separate aggregate, however, this approach has a couple of drawbacks:

- When the user toggles the checkbox, Structure will have to calculate a new aggregate and transfer the results. Because the aggregate values are cached on the server side, and issue data values are cached on the client side, on both sides we'll have increased memory consumption.
- Because of the way the aggregates are calculated and cached on the server side, the aggregate for the option turned off will be somewhat more difficult to write, and use a more complex data structure.

So, we'll do things differently, and use a single, simpler, aggregate, calculating the data with the "Include itself" option turned on. If it's off, we'll adjust the data on the client side. To do that, we'll need another piece of data – the status ID for the current issue, but that can be provided by Structure itself, and the overhead of requiring it is less than that of a separate aggregate.

8.4.1.3 AttributeSpec for Status Bar

Once we understood which attributes will our JavaScript code need, we have to define or find the appropriate attribute specifications for it.

Our status bar is going to be a new attribute, so we need to create an AttributeSpec[616]. 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`.

8.4.1.4 Status Bar Attribute

Now that we know which attribute we need to implement, let's write a loader of that attribute. A loader is an instance of AttributeLoader[617] 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.

616 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/AttributeSpec.html
617 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/loader/AttributeLoader.html

This basic class is "naive" because the way it implements the distinct counting is very basic – and potentially consuming a lot of memory. Structure has more advanced methods for calculating the distinct totals, but it's not a part of the API as of version 17.

As the loader does not have any other parameters, we'll only need a single instance, which we'll keep in a `static final` field.

```
private static final AttributeLoader<Map<String, Integer>> LOADER = new
 StatusBarLoader();
```

Our loader will have a dependency on the `CoreAttributeSpecs.STATUS` attribute. Structure will guarantee that the dependency attributes are loaded before our loader is asked to do its calculation.

All multi-row attribute loaders (aggregates, propagates and scanning loaders) must not access items directly, but rather declare dependency on item-based or single row-based attributes. This is needed by the attributes subsystem to function efficiently, but also, it lets other developers extend the applicability of those dependency attributes to a new type of items. In that way your aggregate attribute will immediately work with the new item types, even though you didn't know about them at development time.

The calculation of the result is pretty straightforward. The base class, `AbstractDistinctAggregateLoader`, defines two methods for building recursive value: `getRowValue()` provides a single value for a single row and `combine()` accumulates the provided values.

- As a result for a single row, we create a map with just one record: the issue's status is mapped to 1. If status is missing (as would be the case for non-issues), we just return null.
- As a combination function we will implement map merge that combines counters.
- Finally, we return an immutable copy of the `result` map.

StatusBarAggregate.java

```
  private static class StatusBarLoader extends
 AbstractNaiveDistinctAggregateLoader<Map<String, Integer>> {
    public StatusBarLoader() {
      super(STATUS_BAR);
    }

    public Set<AttributeSpec<?>> getAttributeDependencies() {
      return Collections.singleton(STATUS);
    }

    protected Map<String, Integer> getRowValue(RowAttributeContext context) {
      Status value = context.getDependencyValue(STATUS);
      return value == null ? null : Collections.singletonMap(value.getId(), 1);
    }

    protected Map<String, Integer> combine(Collection<Map<String, Integer>> values,
AggregateAttributeContext context) {
      HashMap<String, Integer> r = new HashMap<>();
      for (Map<String, Integer> map : values) {
        if (map != null) {
          for (Map.Entry<String, Integer> e : map.entrySet()) {
            Integer count = r.get(e.getKey());
            if (count == null) {
              count = 0;
            }
            r.put(e.getKey(), count + e.getValue());
          }
        }
      }
      return r;
    }

  }
```

8.4.1.5 Attribute Provider

Attribute providers are registered as modules in the plugin descriptor, and their instances are created by the JIRA module system. If the attribute provider "recognizes" the attribute specification and can serve it, it must return a non-null `AttributeLoader` instance. Because our `StatusBarLoader` implementation is stateless and has no parameters, we can reuse the single `static final` instance, but a configurable data provider could create and return new loaders for each call. The returned loader will then be called once for each item needed to display the Structure grid (or its visible part).

StatusBarDataProvider.java

```
public class StatusBarAttributeProvider implements AttributeLoaderProvider {
  private static final AttributeSpec<Map<String, Integer>> STATUS_BAR = new
 AttributeSpec("com.almworks.statusbar", ValueFormat.JSON_OBJECT);
  private static final AttributeLoader<Map<String, Integer>> LOADER = new
 StatusBarLoader();

  public AttributeLoader<?> createAttributeLoader(AttributeSpec<?> attributeSpec,
@NotNull AttributeContext context)
    throws StructureProviderException
  {
    if (STATUS_BAR.getId().equals(attributeSpec.getId())) {
      return LOADER;
    }
    return null;
  }
}
```

When the data provider is ready, we register it in the plugin descriptor.

atlassian-plugin.xml

```
<structure-attribute-loader-provider key="alp-sbcolumn" name="attribute-loader:Status
Bar Column"
                                    class="com.almworks.jira.structure.sbcolumn.Stat
usBarAttributeProvider"/>
```

8.4.1.6 Client-Side Column

We now come to the most visible part of the column – the client-side JavaScript code, responsible for rendering the cells of the Structure grid and showing the column configuration UI. Having almost 400 lines of JavaScript, the code is too long to be reproduced in its entirety. We advise you to download the API examples source code from the API Usage Samples(see page 965) page and open `sbcolumn.js` from the `status-bar-column` sample plugin in your favorite editor.

First, we'll take a high-level overview of the API and look at a few common concepts – column specifications, column context, and the metadata. After that we'll discuss each of the API classes and their implementations.

API Overview

The whole API is accessible through the `window.almworks.structure.api`(see page 944) global object. There are a few utility functions and four main classes that the developer needs to extend (by using the `api.subClass()` function) in order to create a fully-functional column. These classes are linked together by the **column specification**, which is a JSON object representing all of the column's parameters. Column specifications are discussed in detail in the following section. Now let's overview the classes and functions.

Class or Function	Description
`api.ColumnType`(see page 956)	The **column type** is the gateway between Structure and your code. The column type is registered with Structure and has the following responsibilities: • creating column presets for the "Add Column" menu; • creating the column preset used when switching to your column type from a different type; • creating `Column` and `ColumnConfigurator` instances for given column specifications.
`api.Column`(see page 947)	The **column** is responsible for value rendering. It creates the HTML for the widget cells and controls the column's name and width. It can require one or more attributes to be downloaded from the server for the rendered rows.
`api.ColumnConfigurator`(see page 951)	The **configurator** is responsible for the column configuration panel as a whole. Its most important task is to create `ColumnOption` instances.
`api.ColumnOption`(see page 953)	The **option** is the workhorse of the configuration UI, corresponding to a single "row" of the configuration panel. It is responsible for creating the input elements and routing changes between them and the specification.
`api.registerColumnType(columnType, columnKey)`	Registers a column type with the Structure, making it responsible for handling the given column key (see below).
`api.registerColumnGroup(parameters)`	Registers a new group in the Add Column menu.

Column Specifications

A **column specification** is a JSON object representing the complete configuration of a Structure widget column. Column specifications are stored as parts of view specifications. Each `Column`, `ColumnConfigurator` and `ColumnOption` instance has its own current specification, accessed via `this.spec`. A `ColumnType` is given a column specification when Structure wants it to create a `Column` or a `ColumnConfigurator`. `ColumnType` also creates column specifications for column presets. Finally, column specifications are passed to the export renderer providers on the server side (see below).

Do not confuse column specifications with attribute specifications. A column is a higher-level concept and may require multiple attributes (as is the case with our Status Bar column).

Here is an example of a Status Bar column specification.

```
{ "csid":   "7",
  "key":    "com.almworks.jira.structure.sbcolumn",
  "name":   "Status Bar",
  "params": {
    "statuses":      ["1", "3", "4", "5", "6", "10000"],
    "colors":        ["#fcaf3e", "#fce94f", "#ef2929", "#8ae234", "#ad7fa8",
"#729fcf"],
    "includeItself": true }}
```

Key	Description
`csid`	The **CSID** ("column sequential ID") is a string that uniquely identifies a column within a view. CSIDs are assigned and managed by Structure, and should not bother you as a column developer. **Do not change a column's CSID!**
`key`	The **key** is a string identifying the column type. Structure uses the key to decide which `ColumnType` or `ExportRendererProvider` to use for a particular column. The key is required.
`name`	The column **name** is shown in the column header. The name is often omitted from the specification, in which case a default name is generated for the column.
`params`	This is a JSON object containing the column's parameters. The layout of this object is up to the column developer. In the example we see two parallel arrays for the selected status IDs and their colors, and a `boolean` for the "Include itself" option.

The Column Context

A **column context** is a JavaScript object providing various kinds of information about the environment, in which columns and their configurators operate. It is not to be confused with the somewhat similar in purpose, but unrelated `AttributeContext` on the server side. When Structure makes requests to the `ColumnType`, it passes the context as a parameter. Each `Column`, `ColumnConfigurator` or `ColumnOption` instance has its own current context, accessed via `this.context`. The table below describes the methods of the column context.

Method	Description
`structure.isPrimaryPanel()`	Returns `true` if the column belongs (or will belong, for presets) to the primary panel of the Structure widget.

Method	Description
`structure.isSecondaryPanel()`	Returns `true` if the column belongs (or will belong, for presets) to a secondary panel(see page 350) of the Structure widget.
`structure.isStructureBoard()`	Returns `true` if the current widget is on the Structure Board page.
`structure.isIssuePage()`	Returns `true` if the current widget is in the Structure section of an issue page.
`structure.isGadget()`	Returns `true` if the current widget is embedded in a Structure gadget.
`structure.isLocalGadget()`	Returns `true` if the current widget is embedded in a **local** Structure gadget (i.e. a gadget provided and rendered by the same server).
`structure.isRemoteGadget()`	Returns `true` if the current widget is embedded in a **remote** Structure gadget (i.e. a gadget provided and rendered by different servers).
`structure.isGreenHopperTab()`	Returns `true` if the current gadget is in the Structure section of an Agile (GreenHopper) board.
`structure.isProjectPage()`	Returns `true` if the current gadget is in the Structure tab of a project page.
`jira.getAllIssueFields()`	Returns an array of JSON objects representing available JIRA issue fields.
`jira.getIssueFieldById(fieldId)`	Returns a JSON object representing the JIRA issue field with the given ID, or `undefined` if there is no such field.
`getMetadata(key)`	Returns the metadata object associated with the given `key`. See the section below for the description of metadata.

In our column we'll use `context.getMetadata()`.

Requesting and Using Metadata

Metadata, in the context of the column API, is any data needed by column types, columns, and configurators to to do their duties, except for attributes. For example, the Status Bar column needs to know the IDs and names of all the issue statuses in order to render tooltips and create presets – this is metadata. Structure provides some metadata by default – the `getAllIssueFields()` and `getIssueFieldById()` methods of the column context are examples, but you can load more via AJAX by issuing **metadata requests**.

Metadata is requested by overriding one or more of the methods in `ColumnType`, `Column`, and `ColumnConfigurator` classes. Let's look at an example from the Status Bar column type:

sbcolumn.js

```
getMetadataRequests: function() {
  return {
    status: {
      url: baseUrl + '/rest/api/2/status',
      cacheable: true,
      extract: function(response) {
        var result = { order: [], names: {} };
        if ($.isArray(response)) {
          response.forEach(function(status) {
            result.order.push(status.id);
            result.names[status.id] = status.name;
          });
        }
        return result;
      }
    }
  };
}
```

The method is supposed to return a JavaScript object. Each key in that object will become a metadata key for obtaining the corresponding result from the column context. In this example, the status-related metadata object will be obtained by calling `context.getMetadata('status')`.

The values in the returned object are request specifications. Let's look at the request properties:

- The `url` property is the URL to be requested. Here we call a JIRA REST API method that returns all available issue statuses. **Don't forget the JIRA base URL!**
- The `cacheable` property is an opt-in mechanism for response caching. If a metadata request is cacheable, and this URL has already been requested (e.g. by a different column type), the previous response will be used instead of making a new AJAX request. You should **declare your requests cacheable whenever possible** to conserve traffic and improve responsiveness.
- The `extract` property is the function that receives the response and produces the value stored in the metadata map. If omitted, the response is stored unchanged. In the example, we convert the resulting array of JSON objects into an array of status IDs and a map from status IDs to status names.
- You can add any other properties supported by `jQuery.ajax()` to the request specification. Remember, though, that the jQuery success and error handlers will not be called for cacheable requests if a cached response is used.

Different metadata may be required for different operations. Therefore, there are several methods in the API that you can override to request metadata:

- A column type may request metadata to be able to:
  - create column presets – `ColumnType.getPresetMetadataRequests()`;
  - create columns from specifications – `ColumnType.getColumnMetadataRequests()`;
  - create configurators from specifications – `ColumnType.getConfigMetadataRequests()`;
  - do all of the above – `ColumnType.getMetadataRequests()`, the "catch-all" method.
- A column may need metadata to render its values – `Column.getMetadataRequests()`.
- A configurator may need metadata to set up the UI – `ColumnConfigurator.getMetadataRequests()`.

Please note that the corresponding type-level metadata is also available to the columns and configurators created by the type. So, for example, there is no need to issue *the same* requests in both `ColumnType.getColumnMetadataRequests()` and `Column.getMetadataRequests()`, the former alone will suffice.

Structure will delay loading the metadata for as long as possible. For example:

- the metadata for a column will not be loaded unless there is a column in the widget that needs it;
- the metadata for creating column presets will not be loaded until the user clicks "Add Column" or "Edit Column" icons;
- and so on.

Structure guarantees that the metadata request will be completed by the time it calls your type, column, and configurator methods (obviously, except for the `getMetadataRequests()` methods themselves). If the requests succeed, the metadata will be available in the column context. If they fail, the corresponding metadata will be `undefined`, but the methods will still be called, and they should not fail in that case.

`Column`

The `api.Column` class is responsible for rendering the cells of the Structure grid. Please refer to the Column class reference(see page 947) for the list of methods that you can override. The `StatusBarColumn` class in `sbcolumn.js` overrides four methods.

`getDefaultName()` simply returns a localized string as the column name when the name is not present in the column specification. A more involved column could use its specification, context, or metadata to determine the default column name.

`canShrinkWhenNoSpace()` allows Structure to make the column narrower than its minimum width when the widget is very low on horizontal space. Because we do not override any other sizing-related methods, the column will be resizable, with the default and minimum width of 120 and 27 pixels, respectively. Autosizing will not be applied to it, because there is no variable-size content, so autosizing makes no sense.

`collectRequiredAttributes()` always requests the status bar aggregate data from `StatusBarAttributeProvider`. If the "Include itself" option is off, it additionally requests the status ID of the current issue, which is provided by Structure as `{id:'status',format:'id'}`. The main attributes are also available from `require('structure/widget/attributes/Attributes')` object.

`getSortAttribute()` is used to specify the attribute for sorting when the user clicks on the column header.

`getCellViewHtml()` returns the actual HTML for the cells. It obtains the serialized status bar map from the `renderParameters`, deserializes it, adjusts for the "Include itself" option, if necessary, distributes the full status bar width of 100% among the selected statuses according to their issue counts, and finally generates and returns the status bar HTML code as a string. Please refer to the source code for the implementation details.

Please note, that for simple columns, displaying textual information, we advise you to override `getCellValueHtml()` instead, and let Structure take care of the boilerplate HTML surrounding your value.

However, since we want the Status Bar to look similar to Structure's Progress Bar, we need to override a higher-level method and mimic the Progress Bar HTML layout.

ColumnConfigurator

The api.ColumnConfigurator(see page 951) class is responsible for the column configuration UI. Because most of the work is delegated to ColumnOption instances (see below), the configurators themselves are usually quite simple. Let's look at StatusBarConfigurator in its entirety.

sbcolumn.js

```
var StatusBarConfigurator = api.subClass('StatusBarConfigurator',
api.ColumnConfigurator, {
  init: function() {
    this.spec.key = COLUMN_KEY;
    this.spec.params || (this.spec.params = {});
  },
  getColumnTypeName: function() {
    return AJS.I18n.getText("sbcolumn.name");
  },
  getGroupKey: function() {
    return GROUP_KEY;
  },
  getOptions: function() {
    return [new StatusesOption({ configurator: this }), new
 IncludeItselfOption({ configurator: this })];
  }
});
```

The constructor, init() simply sanitizes the current column specification.

getColumnTypeName() returns the human-readable name for the column type. This name is used in the "Type" drop-down of the column configuration panel. You can also override getDefaultColumnName() to generate column names if the type name cannot always be used as the default column name.

getGroupKey() returns the key of the group in the "Add Column" menu that will contain this preset. See the sections on ColumnType(see page 875) and column groups(see page 876) below.

getOptions() creates and returns an array of ColumnOption instances that create input controls for the column configuration panel and route events. Please note how the configurator instance is passed to each option's constructor – this is crucial. The order of the options in the resulting array is also important – the rows of the configuration panel will be created in that order.

Although the methods of StatusBarConfigurator always return the same values, this is not a requirement. The result of any of the methods can depend on the current column specification (this.spec) and metadata.

ColumnOption

Each api.ColumnOption(see page 953) instance is responsible for editing a single logical "part" of the column specification, and corresponds to a single "row" of the column configuration panel. The option creates the actual input elements and sets up event handlers to transfer the values between the inputs and its column specification. An option can hide itself if it's not applicable to the current specification. Also, each option can prohibit saving the

column configuration if it considers the current specification invalid – see `isInputValid()` method in the class reference.

Status Bar column has two options:

- `StatusesOption` is responsible for status selection, colors, and ordering. It "owns" the `statuses` and `colors` arrays of a Status Bar column specification. This option is somewhat more involved than the next one, but you can still refer to its source code in `sbcolumn.js`.
- `IncludeItselfOption` is responsible for the "Include itself" checkbox and "owns" the `includeItself` specification parameter. This is one of the simplest options imaginable, so we'll look at its code in detail.

sbcolumn.js

```
var IncludeItselfOption = api.subClass('IncludeItselfOption', api.ColumnOption, {
  createInput: function(div$) {
    this.checkbox$ = div$.append(
        AJS.template('<div class="checkbox"><label><input type="checkbox"> 
{label}</label></div>')
          .fill({ label: AJS.I18n.getText("sbcolumn.include-itself") })
          .toString()).find('input');
    var params = this.spec.params;
    this.checkbox$.on('change', function() {
      if ($(this).is(':checked')) {
        params.includeItself = true;
      } else {
        delete params.includeItself;
      }
      div$.trigger('notify');
    });
  },
  notify: function() {
    this.checkbox$.prop('checked', !!this.spec.params.includeItself);
    return true;
  }
});
```

Because the option class specifies no `title` and doesn't override `createLabel()`, there is no label to the left of the checkbox.

The `createInput()` method creates the checkbox and sets up event handling. It is passed a jQuery object to append the input elements to.

Please note that Structure column configuration panels use the AUI Forms[618] 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.

618 https://developer.atlassian.com/display/AUI/Forms

`ColumnType`

The `api.ColumnType`(see page 956) class is the main entry point used by the Structure plugin to call your client-side column code. A column type instance creates column presets, columns, and configurators. To find the complete source code for the Status Bar column type, please open `sbcolumn.js` from the API example sources(see page 965) in your favorite editor and scroll to the `StatusBarType` class definition.

The `getMetadataRequests()` method declares the column-level metadata request to load the available issue statuses from JIRA. See Requesting and Using Metadata(see page 871) above for details.

The `createSwitchTypePreset()` method creates a single column specification, which is used as a preset when the user selects our type in the "Type" drop-down on the column configuration panel.

Note the call to the `isAvailable()` function that checks that the preset is needed for the primary panel and that the status metadata is indeed available. If that check fails, the method returns `null`, making it impossible to switch to the Status Bar column type. You can try it yourself – open the Search Result secondary panel, add any column to it and try to change its column type. You should see that the Status Bar type is not available.

The switching preset doesn't have to be fully configured, because the configuration panel is already open when it's used. However, because the Status Bar column configuration is quite complex, we make an extra effort and pre-populate the preset with all the known statuses and some default colors for them. This way the user will quickly see what a status bar looks like without having to configure anything at all. This tactic can be useful for other columns with a lot of parameters.

The `createAddColumnPresets()` method creates an array of column specifications that will be used as presets in the "Add Column" menu. Unlike the "switch" preset above, these presets must be completely configured. Like `createSwitchTypePreset()`, this method calls `isAvailable()` first, so a Status Bar column cannot be added to a secondary Structure panel.

Because the "Add Column" menu is the first place where the user discovers your column type, it would be best if your presets are interesting and cover the whole range of the type's functionality. It's not easy to be creative with the Status Bar column though, unless we know the semantics of statuses, which can be arbitrary. So, for simplicity `StatusBarType` adds only a single preset to the "Add Column" menu, reusing the "switch" preset, which is fully configured.

Besides the usual `key`, `name`, and `params`, the "add" presets can have two special properties:

- `presetName` is a string that specifies the name of the preset in the "Add Column" menu. This name will be used *only in the menu*, the added column will have either the `name` from the specification or the default name generated for it. If omitted, the column name will be used as the preset name.
- `shouldOpenConfigurator` – if this flag is set to `true`, the column configuration panel will open immediately after adding the column with this preset. This can be used to create a "Custom..." kind of preset that lets the user explore the available options.

The `createColumn()` and `createConfigurator()` methods return a `Column` or a `ColumnConfigurator` for the given specification, respectively. The methods are similar – they check whether the type is available and the given specification is valid, and if both checks succeed, they instantiate the appropriate subclass. Please note how the column context and the specification are passed to the constructors, this is crucial.

Finally, at the end of the script we instantiate and register our column type, making it available to Structure:

sbcolumn.js

```
api.registerColumnType(new StatusBarType(), COLUMN_KEY);
```

Structure will use our column type instance to handle the columns with the given key. You can also pass an array of keys as the second argument, to associate your type with more than one column key.

Column Groups

Column groups are used to organize column presets in the "Add Column" menu. Each group has a string key and a human-readable name. Column configurator's `getGroupKey()` method should return the appropriate group key for its preset specification.

Structure specifies four column groups for its built-in columns – `fields`, `icons`, `totals`, and `progress`. For the Status Bar column we will register a separate column group:

sbcolumn.js

```
api.registerColumnGroup({ groupKey: GROUP_KEY, title: AJS.I18n.getText("sbcolumn.name
"), order: 1000 });
```

The `order` parameter determines the position of the group within the menu. The higher the order, the lower the group will be. Structure's predefined groups have order between 100 and 400, inclusive.

Web Resources and Contexts

You need to register your JavaScript and CSS code as a web resource in the plugin descriptor. The Status Bar column has no CSS of its own, and all of its JavaScript code is in a single file, `sbcolumn.js`. Because we use the Structure JavaScript API and the `AJS.template()` function from the Atlassian API, we need to declare two dependencies. We also declare a resource transformation to make `AJS.I18n.getText()` calls work.

atlassian-plugin.xml

```
<web-resource key="wr-sbcolumn" name="web-resource:Status Bar Column">
  <dependency>com.atlassian.auiplugin:ajs</dependency>
  <dependency>com.almworks.jira.structure:widget</dependency>
  <transformation extension="js">
    <transformer key="jsI18n"/>
  </transformation>
  <resource type="download" name="sbcolumn.js" location="js/sbcolumn/sbcolumn.js"/>
  <context>structure.widget</context>
</web-resource>
```

We use `structure.widget` web resource context to make our JavaScript (and CSS, if we had any) load on Structure Board. It also works for the Structure's Dashboard Gadget. However, if you'd like your column to work on other pages – Project page, Issue page or Agile Board page, you need to include other web contexts too – see Loading Additional Web Resources For Structure Widget(see page 879).

8.4.1.7 Export Renderers

Any structure can be exported into printable HTML and Microsoft Excel formats. Exporting is different from rendering the Structure widget in several aspects:

- It is entirely a server-side task, so the code is written in Java.
- The data needed for exporting need not be transferred over the network and cached.
- The export result need not be updated as the exported issues or structure change.
- There are two distinct formats, or media, that are quite different from each other. More formats may be added in the future.

It is because of these differences, that the exporting architecture and APIs are different from their widget rendering counterparts, being simpler in some aspects and more complex in others, while quite similar overall, sometimes making it non-trivial to avoid "repeating yourself".

Please refer to the javadocs[619] 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`[620] is analogous to `AttributeLoaderProvider`, and `ExportRenderer`[621] is a mixture of `AttributeLoader` and the client-side `Column`(see page 947).

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.

619 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/export/package-summary.html
620 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/export/ExportRendererProvider.html
621 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/export/ExportRenderer.html

Generic Renderer Provider

The `StatusBarRendererProvider` class in the `status-bar-column` example plugin source contains both the generic provider and its renderer. The code is quite long, but that's mostly due to defensive checks and the general verbosity of Java. The operation of both the provider and the renderer is quite straight-forward.

The provider's `getColumnRenderer()` method does the following:

- Checks that the given column specification indeed represents a Status Bar column, just in case.
- Obtains the column name form the specification, generating a default name if there is none.
- Extracts the `statuses` array and the `includeItself` flag from the specification parameters. These are needed for rendering.
- Creates and returns an instance of the `StatusBarRenderer` inner class, passing it the column name and parameters.

The renderer has `prepare()` method that lets it specify which attributes it will need loaded to do the export. Like in `StatusBarColumn`, we request our histogram-based custom attribute and status for the current row.

The renderer's `configureColumn()` method sets the column name by calling `setText()` on the given column's header cell.

The renderer's `renderCell()` method does the following:

- Obtains the attribute values from the context.
- Adjusts the data if the the "Include itself" option is off, by decrementing the issue count for the current issue's status.
- Iterates over the selected statuses, adding each non-zero sub-issue count and the corresponding status name to a `StringBuilder`.
- If the resulting value is not empty, calls `setText()` on the given cell.

Here is the module declaration for the generic renderer provider. Note that it specifies the column key, but no export format.

atlassian-plugin.xml

```
<structure-export-renderer-provider key="erp-sbcolumn" name="export-renderer:Status
Bar Column Provider"
                                    class="com.almworks.jira.structure.sbcolumn.Statu
sBarRendererProvider">
  <column-key>com.almworks.jira.structure.sbcolumn</column-key>
</structure-export-renderer-provider>
```

Advanced Excel Renderer Provider

The `StatusBarExcelProvider` class contains the advanced Excel renderer and the corresponding provider.

The provider's `getColumnRenderer()` method is very similar to the generic provider's, with two additions:

- it checks that the export format is indeed `MS_EXCEL`;
- it also extracts the `colors` array from the specification parameters, as the renderer will use those (or similar) colors for the progress bar.

The renderer's `prepare()` and `configureColumn()` methods are the same as the generic version. The `renderCell()` method begins in a similar way, by extracting the data map and adjusting it for the "Include itself" option, if needed.

The interesting part is the actual rendering. The pseudo-graphic "progress bar" that the renderer creates is a string of 30 "pipe" characters, split into colored stripes with lengths proportional to issue counts. `ExcelCell` provides no support for rich text formatting (besides `setRichTextFromHtml()`, which is not up to the task), but we can access the lower-level API, Apache POI HSSF[622], by obtaining the underlying POI objects from `ColumnContext.getObject()`[623] using the keys from `ColumnContextKeys.Excel`[624].

The code that distributes the 30 characters among the stripes is ported from `sbcolumn.js`. To completely understand how the rich text part works, you'll need some knowledge of the POI HSSF API, which is quite complex and outside of the scope of this document. Please refer to the POI documentation and the `StatusBarExcelProvider` source code for more information.

The module declaration for the Excel renderer provider is given below. Note that it specifies both a column key and an export format, thus overriding the generic provider for the Excel format.

atlassian-plugin.xml

```
<structure-export-renderer-provider key="erp-sbcolumn-excel" name="export-
renderer:Status Bar Column Excel Provider"
                                    class="com.almworks.jira.structure.sbcolumn.Statu
sBarExcelProvider">
  <column-key>com.almworks.jira.structure.sbcolumn</column-key>
  <export-format>ms-excel</export-format>
</structure-export-renderer-provider>
```

8.4.2 Loading Additional Web Resources For Structure Widget

To include a web resource (such as custom CSS or JavaScript file) on the page every time Structure Widget(see page 83) is displayed, use `structure.widget` web resource context and possibly a few others.Use cases:

1. You create your own custom field and would like it to be editable in the Structure grid. The field is powered by additional JavaScript or CSS, which should be loaded on the page that displays structure.
2. You create your own column type(see page 862). You'll need to use the Structure JavaScript API(see page 944) and register the web resource with your JavaScript code as a widget extension.

8.4.2.1 Using Web Resource Contexts

You can add JavaScript or CSS to the Structure widget by adding a web resource to the `structure.widget` context. Note, however, that due to Atlassian API limitations, context-provided web resources may not be loaded on **all** pages with the Structure widget. The following table lists all web resource contexts related to pages where Structure Widget can possibly be shown.

Web Resource Context	Used on...
`structure.widget`	Structure Board, Structure Gadget
`jira.view.issue`	Issue Page, Issue Navigator in details view

622 http://poi.apache.org/spreadsheet/index.html
623 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/column/ColumnContext.html#getObject(java.lang.Object)
624 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/column/ColumnContextKeys.Excel.html

Web Resource Context	Used on...
gh-rapid	Scrum and Kanban boards from JIRA Software
jira.browse.project	Project page (including Structure tab)
structure.printable	Printable Structure page

To have your code present on every page where a Structure widget can possibly be shown, include all these resources. You usually don't need to include `structure.printable` though, unless you have some special rules for printing.

Sample snippet from `atlassian-plugin.xml`:

```
<web-resource key="custom-field-resource" name="My Custom Field Web Resource">
  <resource type="download" name="custom-field-resource.js" location="js/myplugin/
custom-field-resource.js"/>
  <context>structure.widget</context>
  <context>jira.view.issue</context>
  <context>gh-rapid</context>
  <context>jira.browse.project</context>
</web-resource>
```

8.4.3 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.

8.4.3.1 Example

```
<structure-item-type key="type-milestone" name="itemtype:Milestone" weight="100"
                     class="com.almworks.jira.structure.api.item.generic.GenericItemT
ype">
 <icon spanClass="my-app-milestone-icon-class"/>
 <displayable key="my.app.milestone.displayable"/>
 <title key="my.app.milestone.title"/>
 <newItemTitle key="my.app.milestone.new"/>
</structure-item-type>
```

Element	Required	Description
`@key`	Yes	Unique module key within the plugin. Full module key will define the `itemType` part of `ItemIdentity`[625].
`@name`	No	A human-readable name of the plugin module.
`@weight`	No	Determines the order in which generic item types appear in menus and lists. Items with the 'lightest' weight are displayed first and the 'heaviest' items sink to the bottom.
`@class`	Yes	Module class. Must be `GenericItemType`[626].
`icon`	No	An icon shown in the item row. At this point the only supported option is using a single icon, associated with a CSS class, for all items of the given type. If you're using your own icons, make sure the appropriate CSS styles are loaded everywhere Structure can be used (see Loading Additional Web Resources For Structure Widget[627]).
`displayable`	Yes	An I18N key used to generate a textual representation of an item for activity streams, decision panels, and elsewhere. The value should contain the type name and a placeholder for the item name, e.g. "milestone {0}".
`title`	Yes	An I18N key for the item creation panel, e.g. "Milestone".
`newItemTitle`	Yes	An I18N key for the Structure toolbar's "Add item" drop-down menu, e.g. "New Milestone".

8.4.3.2 Programmatic Access to Generic Items

Use `GenericItemService`[628] or `GenericItemManager`[629] 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.

625 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/ItemIdentity.html
626 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/generic/GenericItemType.html
627 https://wiki.almworks.com/display/structure2gmaster/Loading+Additional+Web+Resources+For+Structure+Widget
628 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/generic/GenericItemService.html
629 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/generic/GenericItemManager.html

8.4.3.3 Generic Item Permissions

Each generic item is associated with the structure that contains it, and that structure's permissions are used to determine who can see and update the item.

- Any user can create a new generic item programmatically. **Edit** access level is required to add the new item to a structure. When the item is added to a structure, it becomes associated with that structure.
- As with issues and folders, **Edit** access level is required to create a generic item using Structure UI. The item is associated with the structure it was created in.
- If a generic item from one structure is copied or moved to a different structure, a copy of the item is created and associated with the new structure.
- All users having the **View** access level to a structure can view all generic items in that structure.
- All users having the **Edit** access level to a structure can update and delete all generic items in that structure.
- As with any other item, **Edit** access level is required to remove a generic item from a structure. When a generic item is removed from a structure it is **not** deleted from the database. It can still be seen in structure history(see page 375), accessed or updated programmatically, and re-inserted into the structure.

8.5 Accessing Structure Data Remotely

Structure plugin provides REST API, which is primarily used by the structure widget(see page 78). The same API can be used to access the hierarchical data remotely from an automation script or another user agent application.

See details in the REST API Reference(see page 896).

8.6 Reference

8.6.1 Structure Developer Reference

- Structure Java API Reference(see page 883)
  - Structure API Versions(see page 883)
- Structure Plugin Module Types(see page 888)
  - structure-attribute-loader-provider(see page 889)
  - structure-export-renderer-provider(see page 889)
  - structure-item-type(see page 891)
  - Generator Modules(see page 891)
  - structure-effector(see page 893)
  - structure-effect-provider(see page 894)
  - structure-query-constraint(see page 895)
  - new-structure-template(see page 895)
- Structure REST API Reference(see page 896)
  - Structure Resource(see page 897)
  - Forest Resource(see page 917)
  - Item Resource(see page 920)
  - Value Resource(see page 924)
  - Attribute Subscription Resource(see page 928)
  - Configuration Resource(see page 933)
- Structure JavaScript API Reference(see page 944)
  - JavaScript API Functions(see page 944)
  - JavaScript API Classes(see page 946)

        - Column Class(see page 947)
        - ColumnConfigurator Class(see page 951)
        - ColumnOption Class(see page 953)
        - ColumnType Class(see page 956)
        - ItemDetailsProvider Class(see page 958)
    - JavaScript API Objects(see page 962)
        - ItemDetailsBridge Object(see page 962)
- Web Resource Contexts(see page 965)

8.6.2 Structure Java API Reference

> Structure API is work in progress. You will find that some of the packages are documented less than others, and some are not documented yet.
>
> We're constantly working on the API improvements and documentation and will make the javadocs and other parts of the documentation more complete with every release.

Structure API Reference for the latest version: http://almworks.com/structure/javadoc/latest

You can download javadocs from the Maven repositories into your IDE.

Check out information about Structure API Versions(see page 883) to select the correct API artifact, and you can also download Javadoc JARs there.

8.6.2.1 Structure API Versions

Current Versions

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
17.6.0 Javadocs[630]	Jira 7.13+	6.4.0+	"[17.6,18)"	2020-12-01
17.5.0 Javadocs[631]	Jira 7.13+	6.3.2+	"[17.5,18)"	2020-11-18

630 http://almworks.com/structure/javadoc/17.6.0
631 http://almworks.com/structure/javadoc/17.5.0

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
17.4.0 Javadocs[632]	Jira 7.13+	6.3.0+	"[17.4,18)"	2020-10-08
17.3.0 Javadocs[633]	Jira 7.13+	6.2.0+	"[17.3,18)"	2020-08-10
17.2.0 Javadocs[634]	Jira 7.13+	6.1.0+	"[17.2,18)"	2020-06-03
17.1.0 Javadocs[635]	Jira 7.13+	6.0.0+	"[17.1,18)"	2020-03-20
17.0.0 Javadocs[636]	Jira 7.13+	6.0.0.eap1+	"[17,18)"	2020-01-30
16.16.0 Javadocs[637]	Jira 7.6+	5.6.0+	"[16.16,17)"	2019-09-18

632 http://almworks.com/structure/javadoc/17.4.0
633 http://almworks.com/structure/javadoc/17.3.0
634 http://almworks.com/structure/javadoc/17.2.0
635 http://almworks.com/structure/javadoc/17.1.0
636 http://almworks.com/structure/javadoc/17.0.0
637 http://almworks.com/structure/javadoc/16.16.0

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
16.15.0 Javadocs[638]	Jira 7.6+	5.5.0+	"[16.15,17)"	2019-07-08
16.14.0 Javadocs[639]	Jira 7.6+	5.4.0+	"[16.14,17)"	2019-06-03
16.13.0 Javadocs[640]	Jira 7.6+	5.3.0+	"[16.13,17)"	2019-03-22
16.12.0 Javadocs[641]	Jira 7.6+	5.2.0+	"[16.12,17)"	2018-12-25
16.11.0 Javadocs[642]	Jira 7.2+	5.1.0+	"[16.11,17)"	2018-10-25
16.10.0 Javadocs[643]	Jira 7.2+	5.0.0+	"[16.10,17)"	2018-08-16

638 http://almworks.com/structure/javadoc/16.15.0
639 http://almworks.com/structure/javadoc/16.14.0
640 http://almworks.com/structure/javadoc/16.13.0
641 http://almworks.com/structure/javadoc/16.12.0
642 http://almworks.com/structure/javadoc/16.11.0
643 http://almworks.com/structure/javadoc/16.10.0

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
16.9.0 Javadocs[644]	Jira 7.2+	4.6.0+	"[16.9,17)"	2018-03-28
16.8.0 Javadocs[645]	Jira 7.2+	4.5.0+	"[16.8,17)"	2017-12-26
16.7.0 Javadocs[646]	Jira 7.2+	4.4.0+	"[16.7,17)"	2017-11-29
16.6.0 Javadocs[647]	Jira 7.2+	4.3.0+	"[16.6,17)"	2017-10-16
16.5.0 Javadocs[648]	Jira 7.2+	4.2.0+	"[16.5,17)"	2017-08-25
16.4.0 Javadocs[649]	Jira 7.1+	4.1.0+	"[16.4,17)"	2017-06-19
16.3.0 Javadocs[650]	Jira 7.1+	4.0.0+	"[16.3,17)"	2017-04-26
16.2.0 Javadocs[651]	Jira 7.0+	3.6.0+	"[16.2,17)"	2017-04-03

644 http://almworks.com/structure/javadoc/16.9.0
645 http://almworks.com/structure/javadoc/16.8.0
646 http://almworks.com/structure/javadoc/16.7.0
647 http://almworks.com/structure/javadoc/16.6.0
648 http://almworks.com/structure/javadoc/16.5.0
649 http://almworks.com/structure/javadoc/16.4.0
650 http://almworks.com/structure/javadoc/16.3.0
651 http://almworks.com/structure/javadoc/16.2.0

Version	Supported Jira Versions	Supported Structure Versions	OSGi Import Version	Release Date
16.1.0 Javadocs[652]	Jira 7.0+	3.5.0+	"[16.1,17)"	2017-01-26
16.0.0 Javadocs[653]	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[654]*.*				

Javadocs for the Latest Version[655] — Java API documentation for the latest API version.

To see how to include the API in your project dependencies, read about Accessing Structure from Jira Plugin(see page 845).

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-17.6.0.jar -DpomFile=structure-api-17.6.
0.pom
```

File	Modified
structure-api-17.6.0-javadoc.jar[656]	Dec 07, 2020 by Julia Atlygina[657]

652 http://almworks.com/structure/javadoc/16.1.0
653 http://almworks.com/structure/javadoc/16.0.0
654 https://wiki.almworks.com/display/structure0211/Structure+API+Versions
655 http://almworks.com/structure/javadoc/latest
656 https://wiki.almworks.com/download/attachments/32223472/structure-api-17.6.0-javadoc.jar?api=v2
657 https://wiki.almworks.com/display/~atlygina

File	Modified
structure-api-17.6.0-sources.jar[658]	Dec 07, 2020 by Julia Atlygina[659]
structure-api-17.6.0.pom[660]	Dec 07, 2020 by Julia Atlygina[661]
structure-api-17.6.0.jar[662]	Dec 07, 2020 by Julia Atlygina[663]
structure-api-16.16.0-javadoc.jar[664]	Sep 19, 2019 by Julia Atlygina[665]
structure-api-16.16.0-sources.jar[666]	Sep 19, 2019 by Julia Atlygina[667]
structure-api-16.16.0.jar[668]	Sep 19, 2019 by Julia Atlygina[669]
structure-api-16.16.0.pom[670]	Sep 19, 2019 by Julia Atlygina[671]

8.6.3 Structure Plugin Module Types

The following module types are added by the Structure plugin:

- structure-attribute-loader-provider(see page 889) lets you provide the data for new column types in the Structure widget.
- structure-export-renderer-provider(see page 889) lets you export new column types to printable HTML and Excel files.
- structure-item-type(see page 891) lets you define a new type of items, which can be used in structures.
- Generator Modules(see page 891) let you add generators to the Automation subsystem. The following module types are included:
  - `structure-inserter`
  - `structure-extender`
  - `structure-filter`
  - `structure-grouper`
  - `structure-sorter`
- structure-effector(see page 893) lets you add effectors to the Automation subsystem.
- structure-effect-provider(see page 894) lets you define a new effect provider, which can be used by effectors.
- new-structure-template(see page 895) lets you add templates for new structures.
- structure-query-constraint(see page 895) allows adding new functions to S-JQL language.

---

658 https://wiki.almworks.com/download/attachments/32223472/structure-api-17.6.0-sources.jar?api=v2
659 https://wiki.almworks.com/display/~atlygina
660 https://wiki.almworks.com/download/attachments/32223472/structure-api-17.6.0.pom?api=v2
661 https://wiki.almworks.com/display/~atlygina
662 https://wiki.almworks.com/download/attachments/32223472/structure-api-17.6.0.jar?api=v2
663 https://wiki.almworks.com/display/~atlygina
664 https://wiki.almworks.com/download/attachments/32223472/structure-api-16.16.0-javadoc.jar?api=v2
665 https://wiki.almworks.com/display/~atlygina
666 https://wiki.almworks.com/download/attachments/32223472/structure-api-16.16.0-sources.jar?api=v2
667 https://wiki.almworks.com/display/~atlygina
668 https://wiki.almworks.com/download/attachments/32223472/structure-api-16.16.0.jar?api=v2
669 https://wiki.almworks.com/display/~atlygina
670 https://wiki.almworks.com/download/attachments/32223472/structure-api-16.16.0.pom?api=v2
671 https://wiki.almworks.com/display/~atlygina

- structure-synchronizer(see page 974) defines a new synchronizer.

8.6.3.1 structure-attribute-loader-provider

You can use this module to add your support for attributes, either new or already existing, to Structure. The attributes are used by Structure Widget columns, by exporters and by generators.

Example

```
<structure-attribute-loader-provider key="provider-key"
    class="com.company.your.plugin.attribute.MyAttributeProvider"/>
```

Element	Required?	Description
`structure-attribute-loader-provider`	Yes	The module descriptor.
`@key`	Yes	The unique identifier of the plugin module.
`@name`	No	The human-readable name of the plugin module.
`@order`	No	Takes an integer number (including negative numbers) that will define the order in which attribute specs are offered to the providers.
`@class`	Yes	The class that implements the data provider. Must implement AttributeLoaderProvider[672].

8.6.3.2 structure-export-renderer-provider

Export renderer provider module lets you register the components responsible for exporting Structure columns to printable HTML and Microsoft Excel formats.

672 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/loader/AttributeLoaderProvider.html

Export renderer provider example

```
<structure-export-renderer-provider
     key="erp-sbcolumn-excel"
     name="export-renderer:Status Bar Column Excel Provider"
     class="com.almworks.jira.structure.sbcolumn.StatusBarExcelProvider">

  <column-key>com.almworks.jira.structure.sbcolumn</column-key>
  <export-format>ms-excel</export-format>
</structure-export-renderer-provider>
```

Element	Required?	Description
`structure-export-renderer-provider`	Yes	The module descriptor.
`@key`	Yes	The unique identifier of the plugin module.
`@name`	No	The human-readable name of the plugin module.
`@class`	Yes	The class that implements the renderer provider. Must implement ExportRendererProvider[673].
`column-key`	No	The column key that this provider is associated with. You can have multiple `column-key` elements in a single descriptor. If no column key is specified, the renderer provider is considered generic – such a provider will be consulted for every column not served by a type-specific provider.
`export-format`	No	The export format that this provider is associated with. The values are `printable` for the printable HTML format and `ms-excel` for the Microsoft Excel XLS format. You can have multiple `export-format` elements in a single descriptor. If no export format is specified, the renderer provider is considered generic – such a provider will be consulted for every column not served by a format-specific provider.

673 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/export/ExportRendererProvider.html

8.6.3.3 structure-item-type

This module type lets you declare a new item type. Items of that type can then be used in structures.

Example

```
<structure-item-type key="type-book" name="itemtype:Book"
    class="com.mycompany.structure.books.BookItemType"/>
```

Element	Required?	Description
`@key`	Yes	The unique identifier of the plugin module. Full module key will define the `itemType` part of ItemIdentity[674].
`@name`	No	The human-readable name of the plugin module.
`@class`	Yes	The class that implements the support for the item type. Must implement StructureItemType[675].

8.6.3.4 Generator Modules

There are five modules, one for each type of generators, that work in the same way:

- `structure-inserter`
- `structure-extender`
- `structure-filter`
- `structure-grouper`
- `structure-sorter`

Each module allows declaring a generator of a specific type. When a plugin with a generator module is installed, you get the ability add those generators to structures.

674 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/ItemIdentity.html
675 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/StructureItemType.html

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(see page 879)).

You can also use the standard icons used by bundled generators:

Generator Type	Icon Classes
Inserter	`s-fa s-fa-plus`
Extender	`s-fa s-fa-link`
Filter	`alm alm-group`
Grouper	`s-fa s-fa-filter`
Sorter	`alm alm-sort-asc`

8.6.3.5 structure-effector

You can use this module to add a new effector to Structure. Effectors are used to update items based on Structure forests and attribute values.

Example

```
<structure-effector
    key="effector-example" name="effector:Example"
    class="com.mycompany.structure.examples.ExamplesEffector">
  <label key="com.mycompany.examples.effector.label"/>
  <dialog-title key="com.mycompany.examples.effector.dialog-title"/>
  <resource type="velocity" name="form" location="/templates/example/effector-
example-form.vm"/>
  <resource type="velocity" name="summary" location="/templates/example/effector-
example-summary.vm"/>
</structure-extender>
```

Element	Required?	Description
`@key`	Yes	The unique identifier of the effector. Full module key will be a part of effector specification, defining the automation.
`@name`	No	The name of the module for the Jira administrator.

Element	Required?	Description
`@description`	No	Description of the module for the Jira administrator.
`@class`	Yes	Fully-qualified class name of the effector implementation. Must implement Effector[676].
`label`	Yes	The name of the effector as it appears to the user.
`dialog-title`	No	The title of the dialog that is used to edit the effector.
`resource[@name=form]`	No	Velocity template with the HTML form that will be used for editing the effector's parameters.
`resource[@name=summary]`	No	Velocity template that will be used to display the effector as a row in a structure.

8.6.3.6 structure-effect-provider

You can use this module to add your new effect provider to Structure. Effect providers are used by effectors, e.g. to update issue field values.

Example

```
<structure-effect-provider key="provider-key"
    class="com.company.your.plugin.MyEffectProvider"/>
```

Element	Required?	Description
`structure-effect-provider`	Yes	The module descriptor.
`@key`	Yes	The unique identifier of the plugin module.
`@name`	No	The human-readable name of the plugin module.
`@class`	Yes	The class that implements the effect provider. Must implement EffectProvider[677].

676 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effector/Effector.html
677 https://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/effect/EffectProvider.html

8.6.3.7 structure-query-constraint

Structure Query Constraint module allows you to define an additional constraint function that can be used in S-JQL.

For example, `folder()` function explained in S-JQL Reference(see page 272) is implemented with a `structure-query-constraint`.

Example:

```
<structure-query-constraint key="constraint-foo"
                            class="com.mycompany.structure.FooConstraint"
                            name="Structure Query Constraint: foo"
                            fname="foo"/>
```

Element	Required?	Description
key	Yes	Module key.
name	No	Module name for the JIRA administrator.
class	Yes	Class that implements StructureQueryConstraint[678].
fname	Yes	Function name, must be unique throughout the system.

8.6.3.8 new-structure-template

New Structure Template module allows you to add templates to the Create Structure dialog.

Example:

```
<new-structure-template key="big-template"
    class="com.mycompany.structure.template.bigtemplate"
    name="New Structure Template: Big Template">
  <label key="com.mycompany.template.big-template.label"/>
  <description key="com.mycompany.template.big-template.description"/>
  <resource type="download" name="icon.png" location="css/structure/templates/
big@2x.png"/>
  <resource type="velocity" name="step1" location="templates/structure/big/step1.vm"/
>
  <resource type="velocity" name="step2" location="templates/structure/big/step2.vm"/
>
</new-structure-template>
```

678 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/query/StructureQueryConstraint.html

Element	Required?	Description
`@key`	Yes	Module key.
`@name`	No	The name of the module for JIRA administrators.
`@class`	Yes	The class that implements the template, must implement NewStructureTemplate[679].
`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.

8.6.4 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[680] if you'd like to contribute or get preliminary access to that API.

8.6.4.1 General Notes

API Versions

As of Structure version 3.4, there are two versions of the REST API – `1.0` and `2.0`. Some of the REST resources are exposed through version 1.0 and some through version 2.0.

Version 1.0 is stable and we don't plan to change it. It comes from Structure 2 and largely remains the same as in Structure 2.x versions. Some of the resources may become deprecated as we replace them with the newer versions.

Version 2.0 is not stable and is being developed along with the product. That means that you can use it, but you need to test your integration every time you upgrade. We are also going to publish API changes in the release notes.

679 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/template/NewStructureTemplate.html
680 mailto:support@almworks.com

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.

8.6.4.2 REST Resources

- Structure Resource(see page 897) is used to create and manage structures (but not the content)
- Forest Resource(see page 917) is used to retrieve and update forests (a structure's content)
- Item Resource(see page 920) is used to create and update items (issues, folders and, possibly, items of other types)
- Value Resource(see page 924) is used to retrieve attribute values for a given forest
- Attribute Subscription Resource(see page 928) allows subscribing to a particular set of values identified by attributes and rows, and receive updates
- Configuration Resource(see page 933) lets you manage global permissions and the list of projects for which Structure is enabled

8.6.4.3 Structure Resource

This page describes resources with which you can list(see page 902), create(see page 907), read(see page 910), update(see page 912), and delete(see page 916) structures. Structures contain general information(see page 359) such as name and permissions, but not the hierarchy itself. Issue hierarchy is accessed through the Forest Resource(see page 917). This page also documents structure shape(see page 898) and its fields(see page 899), and the error entity(see page 902) that may be returned in case of the REST API user error.

Structure resource belongs to **version 2.0** of the API.

`/structure/ GET`(see page 902)	list structures

`/structure/ POST`(see page 907)	create a structure
`/structure/{id} GET`(see page 910)	read structure
`/structure/{id}/update POST`(see page 912)	update one or several structure fields
`/structure/{id} DELETE`(see page 916)	delete structure

Quick navigation:

- Structure Representations(see page 898)
- Structure Fields(see page 899)
- Permission Rules(see page 900)
- Error Entity(see page 902)

Structure Representations

Structure is represented via JSON. All resources are also capable of producing XML.

```
{
  "id": 103,
  "name": "Structure with all fields",
  "description": "Voilà! This structure exhibits all fields.",
  "readOnly": "true",
  "editRequiresParentIssuePermission": true,
  "permissions": [
    {
      "rule": "apply",
      "structureId": 102
    },
    {
      "rule": "set",
      "subject": "group",
      "groupId": "jira-developers",
      "level": "edit"
    },
    {
      "rule": "set",
      "subject": "projectRole",
      "projectId": 10010,
      "roleId": 10020,
      "level": "admin"
    },
    {
      "rule": "set",
      "subject": "anyone",
      "level": "view"
    },
    {
      "rule": "set",
      "subject": "user",
      "username": "agentk",
      "level": "none"
    }
  ],
  "owner": "user:admin"
}
```


Top(see page 897)

Structure Fields

Structure objects accessible through these resources have the following fields, most of which represent structure details as outlined in the Structure User's Guide(see page 359):

`id`	The ID of the structure (integer, `1..2^63 - 1`.)

`name`	The name of the structure. A structure must have a non-empty name, which does not have to be unique.
`description`	The description on the structure. May be absent.
`readOnly`	`true` if the user has only View(see page 362) access level to the structure, otherwise absent.
`editRequiresParentIssuePermission`	`true` if the Require Edit Issue Permission on Parent Issue(see page 364) flag is set on this structure, otherwise absent.
`permissions`	The list of structure permission rules(see page 363). Present only if the user has Control(see page 362) access level to the structure. Some resources do not include permissions unless requested to do so. List order is as important as the rules themselves.
`owner`	The owner(see page 359) of the structure. Present only if the user is the owner of this structure or if he has Browse Users[681] permission. A string of the form `user:USERNAME`, where USERNAME is the JIRA user login name. Example: `user:jsmith`.

Please note that structure resources described on this page do not include information about issue hierarchies. The content of a structure, i.e. its hierarchy of items, can be read or modified using Forest Resource(see page 917).

Top(see page 897)

Permission rules

There are two types of permission rules, those that *set* permissions and those that *apply* permissions from another structure. They have different fields depending on the type.

Set rules

`rule`	Must be equal to `set`, case-insensitive.
`subject`	Identifies the type of the subject to which the rule applies. Must be one of `group, projectRole, user, anyone`. See below how to identify the subject.

681 https://confluence.atlassian.com/display/JIRA/Managing+Global+Permissions

`level`	Access level to set to the specified subject. Must be equal to one of the names of the Permission Level[682] enum constants, case-insensitive. Please note that Control permission is represented by the ADMIN[683] enumeration constant.

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: `jira-developers`.

REST API user can create such rule only for a group he belongs to.

`projectRole`
The rule applies to all users that have a role in a project.

`projectId`	The ID of the project. Example: `10010`.
`roleId`	The ID of the role. Example: `10010`.

REST API user can create such rule only for roles in projects where Structure is enabled, and for which he has Browse Projects[684] *permission.*

`user`
The rule applies to the user.

`username`	Name of the user. Example: `jsmith` for user John Smith.

REST API user can create such rule only if he has Browse Users[685] *permission, and if such user exists.*

`anyone`
The rule applies to all users, even anonymous (not authenticated.) The rule shouldn't have any additional fields.

Apply rules

`rule`	Must be equal to `apply`, case-insensitive.
`structureId`	The ID of the structure which permissions should be applied. Example: `112`

Apply rule creates a dependency on another structure. Circular dependencies are not allowed. Also, a REST API user can create such rule only if he has Control(see page 362) *access level to the referenced structure.*

Top(see page 897)

682 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html
683 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html#ADMIN
684 https://confluence.atlassian.com/display/JIRA/Managing+Project+Permissions
685 https://confluence.atlassian.com/display/JIRA/Managing+Global+Permissions

Error entity

```
{
  "code": 4005,
  "error": "STRUCTURE_NOT_EXISTS_OR_NOT_ACCESSIBLE[4005]",
  "structureId": 160,
  "message": "Referenced structure [160] does not exist or you don't have Control
permissions on it.",
  "localizedMessage": "Das Struktur [160] existiert nicht oder sie haben keine
Kontrolle Berechtigungen."
}
```

In some cases, requests to structure resources result in an error response containing an error entity. Any of its fields may be absent.

`code`	Integer code of the error
`error`	Brief technical description of the error. Contains a name of the corresponding StructureError[686] enum constant.
`structureId`	The ID of the structure involved.
`issueId`	The ID of the JIRA issue involved.
`message`	More detailed message, may contain technical details.
`localizedMessage`	User-displayable message in the REST API user locale or JIRA default locale if the user is not authenticated.

Top(see page 897)

Structure Resources

GET /structure

```
GET $baseUrl/rest/structure/2.0/structure
GET $baseUrl/rest/structure/2.0/structure?
name=$name&permission=$permission&withPermissions=$withPermissions&withOwner=$withOwn
er&limit=100
```

686 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/StructureError.html

A list of all structures visible to the REST API user. Optionally, the result can be filtered by name or user's access level. By default, permission rules and owners are not included, you should use query parameters if you want them to be included.

> **Who can access this resource**
>
> All users who have access to the Structure Plugin(see page 478). The returned list contains only structures to which the REST API user has at least View(see page 362) access level.

Request
Query parameters:

`name`	If present, the returned list will contain only structures which names contain the specified string (case insensitive).
`permission`	If present, the returned list will contain only structures to which the REST API user has the specified access level(see page 0). Must be equal to one of the names of the Permission Level[687] enum constants, case-insensitive. NONE[688] is treated in the same way as VIEW[689]. Please note that Control permission is represented by the ADMIN[690] enumeration constant.
`withPermissions`	If `true`, permission rules will be included in the response. Default is `false`.
`withOwner`	If `true`, owner will be included in the response. Default is `false`.
`archived`	If `true`, the returned list can also contain archived structures. Default is `false`.
`limit`	If specified, must be a number. Defines the maximum number of structures to return.

Each of the filter parameters `name`, `permission`, or `issueId` can be specified only once, otherwise the first is used. Different parameters are combined with AND.

HTTP headers:

`Content-Type`	Should be one of `application/json`, `application/xml`.

687 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/permissions/PermissionLevel.html
688 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html#NONE
689 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html#VIEW
690 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/PermissionLevel.html#ADMIN

`Accept`	Should be one of `application/json`, `application/xml`.

Response
Success

`200 OK`	Response entity contains the only field, `structures`, which contains the list of the structure objects, sorted by name.	`application/json`, `application/xml`

Example 1: all structures

```
GET $baseUrl/rest/structure/2.0/structure
```


```
{
  "structures": [
    {
      "id": 1,
      "name": "Global Structure",
      "description": "Initial general-purpose structure.",
      "editRequiresParentIssuePermission": true
    },
    {
      "id": 102,
      "name": "Test plan",
      "description": "Test plan #3",
      "readOnly": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2"
    }
  ]
}
```


Example 2: only "Test plan"

```
GET $baseUrl/rest/structure/2.0/structure?name=test+plan
```

```
{
  "structures": [
    {
      "id": 102,
      "name": "Test plan",
      "description": "Test plan #3",
      "readOnly": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2"
    }
  ]
}
```

Example 3: structures that the user can edit with permissions and owners shown

```
GET $baseUrl/rest/structure/1.0/structure?permission=edit&withPermissions=true&withOw
ner=true
```

```
{
  "structures": [
    {
      "id": 1,
      "name": "Global Structure",
      "description": "Initial general-purpose structure.",
      "editRequiresParentIssuePermission": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1",
      "permissions": [
        {
          "rule": "set",
          "subject": "group",
          "groupId": "jira-users",
          "level": "edit"
        },
        {
          "rule": "set",
          "subject": "projectRole",
          "projectId": 10010,
          "roleId": 10010,
          "level": "none"
        },
        {
          "rule": "apply",
          "structureId": 101
        }
      ],
      "owner": "user:jsmith"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2",
      "owner": "user:admin"
    }
  ]
}
```

Example 4: require XML representation
Note that the same can be achieved by specifying `application/xml` in the `Accept` HTTP header.

```
GET $baseUrl/rest/structure/1.0/structure.xml?name=test+plan
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<structureList>
  <structures>
    <structure>
      <id>100</id>
      <name>Test plan</name>
      <description>Test plan #1</description>
    </structure>
  </structures>
</structureList>
```

Error

`400 Bad Request`	`permission` parameter is set to an unknown value, or request is invalid for other reasons. In the first case, response contains error entity, in the second it is empty.	`application/json`, `application/xml`
`403 Forbidden`	If Structure Plugin is not accessible to the REST API user, or if issue with ID `issueId` does not exist or the REST API user does not have enough permissions to access it. Response contains error entity.	`application/json`, `application/xml`
`404 Not Found`	If `issueId` is not an integer. Response entity contains a standard JIRA error HTML page.	`text/html`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 485) operation may be in progress.	

Other return codes[691] are possible under the normal rules of HTTP communication.

Top(see page 897)

POST /structure

```
POST $baseUrl/rest/structure/2.0/structure
```

Create(see page 365) an empty structure by POSTing to this resource.

691 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

Who can access this resource

Only logged in users who have access to the Structure Plugin(see page 478) and a permission to create structures(see page 479).

Request
Request entity should contain the new structure(see page 899). Structure name, `name`, must be present and non-empty. Fields `id`, `readOnly`, and `owner` are ignored. All rules in `permissions` are validated according to their respective rule types(see page 900).

Please note that this resource accepts only JSON structure representation.

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response
Success

`201 Created`	Response entity contains the created structure with fields, including `permissions` and `owner`.	`application/json`, `application/xml`

Example 1: minimal structure

```
POST $baseUrl/rest/structure/2.0/structure
```

Request entity	Response entity
`{ "name":"Test plan" }`	`{ "id": 104, "name": "Test plan", "description": "", "permissions": [], "owner": "user:admin" }`

Example 2: structure with some permissions

```
POST $baseUrl/rest/structure/2.0/structure
```

Request entity	Response entity

{
"name":"Structure with some permissions",
"editRequiresParentIssuePermission": "true",
"permissions":[
{
"rule":"apply",
"structureId":102
}
]
}
	{
	"id": 105,
	"name": "Structure with some permissions",
	"description": "",
	"editRequiresParentIssuePermission": true,
	"permissions": [
	{
	"rule": "apply",
	"structureId": 102
	}
	],
	"owner": "user:admin"
	}

Error

`400 Bad Request`	Structure data is not well-formed (syntax error) or invalid (semantic error.) Not well-formed structure data examples: request JSON is syntactically incorrect; JSON contains unknown field; `name` is not present or empty; `permissions` list contains a *set* rule with `level` set to an invalid value. Invalid structure example: `permissions` list contains a rule that fails validation. Response entity contains error. Problems with *apply* rule usually have `structureId` to indicate the invalid reference.	`application/json`, `application/xml`
`403 Forbidden`	If REST API user is not logged in or does not have permissions to access Structure Plugin or to create structures. Response contains error entity.	`application/json`, `application/xml`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 485) operation may be in progress.	

Other return codes[692] are possible under the normal rules of HTTP communication.

Top(see page 897)

GET /structure/{id}

```
GET $baseUrl/rest/structure/2.0/structure/$id
GET $baseUrl/rest/structure/2.0/structure/$id?
withPermissions=$withPermissions&withOwner=$withOwner
```

This resource allows to obtain structure details(see page 899) for the particular structure. By default, `permissions` and `owner` are not included, use query parameters to include them.

> **Who can access this resource**
>
> All users who have access to the Structure Plugin(see page 478). To access the particular structure, the user has to have at least View(see page 362) access level.

Request
Path parameter:

`id`	the ID of the structure

Query parameters:

`withPermissions`	If `true`, permission rules will be included in the response. Default is `false`.
`withOwner`	If `true`, owner will be included in the response. Default is `false`.

HTTP headers:

`Content-Type`	Should be one of `application/json`, `application/xml`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response
Success

692 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

200 OK	Response entity contains the created structure along with all of its fields. Field `permissions` is included if the REST API user has Control(see page 362) permission on this structure. Field `owner` is included if the REST API user is either the owner of this structure or has Browse Users permission[693].	`application/json`, `application/xml`

Example 1: retrieve structure with ID 100 without permissions and owner

```
GET $baseUrl/rest/structure/2.0/structure/100
```

```
{
  "id": 100,
  "name": "Test plan",
  "description": "Test plan #1"
}
```

Example 2: permissions and owner are requested to be included, but only owner is shown, because the user has only View access as indicated by `readOnly`

```
GET $baseUrl/rest/structure/2.0/structure/102?withOwner=true&withPermissions=true
```

```
{
  "id":102,
  "name":"Test plan",
  "description":"Test plan #3",
  "readOnly":true,
  "owner":"user:admin"
}
```

Example 3: XML representation may be requested in the request URL instead of the `Content-Type` HTTP header

```
GET $baseUrl/rest/structure/2.0/structure/102.xml
```

```
<structure>
  <id>102</id>
  <name>Test plan</name>
  <description>Test plan #3</description>
  <readOnly>true</readOnly>
</structure>
```

Error

693 https://confluence.atlassian.com/display/JIRA/Managing+Global+Permissions

`400 Bad Request`	One of the query parameters is too long.	
`403 Forbidden`	If REST API user does not have permissions to access Structure Plugin or does not have at least View(see page 362) permission on this structure. Response contains error entity.	`application/json`, `application/xml`
`404 Not Found`	If `id` is not an integer in `1..2^63-1`. Response entity contains a standard JIRA error HTML page.	`text/html`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 485) operation may be in progress.	

Other return codes[694] are possible under the normal rules of HTTP communication.

Top(see page 897)

POST /structure/{id}/update

```
POST $baseUrl/rest/structure/1.0/structure/$id/update
```

Update one or several fields of a structure by POSTing to this resource.

> **Who can access this resource**
>
> Only logged in users who have access to the Structure Plugin(see page 478) and Control(see page 362) permission on this structure.

Request
Request entity should contain those structure fields(see page 899) 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.

694 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

If `permissions` field is present, all rules are validated according to their respective rule types(see page 900).

Please note that this resource accepts only JSON structure representation.

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response
Success

`200 OK`	Response entity contains the updated structure with all fields, including `permissions` and `owner`.	`application/json`, `application/xml`

Example 1: change description of the Global Structure

```
POST $baseUrl/rest/structure/1.0/structure/1/update
```

Request entity	Response entity

{
  "description":"Company-wide structure providing the Big Picture."
}
	{
	  "id":1,
	  "name":"Global Structure",
	  "description":"Company-wide structure providing the Big Picture.",
	  "editRequiresParentIssuePermission":true,
	  "permissions":[
	  {
	  "rule":"set",
	  "subject":"anyone",
	  "level":"view"
	  },
	  {
	  "rule":"set",
	  "subject":"group",
	  "groupId":"jira-users",
	  "level":"edit"
	  },
	  {
	  "rule":"set",
	  "subject":"group",
	  "groupId":"jira-administrators",
	  "level":"admin"
	  }
	  ]
	}

Example 2: changing permission rules

```
POST $baseUrl/rest/structure/1.0/structure
```

Request entity	Response entity

{
  "permissions":[
  {
  "rule":"set",
  "subject":"group",
  "groupId":"jira-users",
  "level":"edit"
  },
  {
  "rule":"apply",
  "structureId":101
  }
  ]
}
	{
	  "id": 105,
	  "name": "Structure with some permissions",
	  "description": "",
	  "editRequiresParentIssuePermission": true,
	  "permissions": [
	  {
	  "rule": "set",
	  "subject": "group",
	  "groupId": "jira-users",
	  "level": "edit"
	  },
	  {
	  "rule": "apply",
	  "structureId": 101
	  }
	  ],
	  "owner": "user:admin"
	}

Error

`400 Bad Request`	Structure data is not well-formed (syntax error) or invalid (semantic error.) Not well-formed structure data examples: request JSON is syntactically incorrect; JSON contains unknown field; `permissions` list contains a *set* rule with `level` set to an invalid value. Invalid structure example: `permissions` list contains a rule that fails validation. Response entity contains error. Responses to problems with an *apply* rule usually have `structureId` to indicate the invalid reference.	`application/json`, `application/xml`.
`403 Forbidden`	If REST API user is not logged in, does not have permissions to access Structure Plugin, or does not have Control(see page 362) access level to this structure. Response contains error entity.	`application/json`, `application/xml`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	

`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 485) operation may be in progress.	

Other return codes[695] are possible under the normal rules of HTTP communication.

Top(see page 897)

DELETE /structure/{id}

Deletes(see page 374) the designated structure.

> **Who can access this resource**
>
> Only logged in users who have access to the Structure Plugin(see page 478) and Control(see page 362) permission on this structure.

Request
Path parameter:

`id`	the ID of the structure

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be absent or equal to one of `application/json`, `application/xml`.

Response
Success

`200 OK`	Contains an object with the only field `empty` with value `true`.	`application/json`, `application/xml`

Note: it should have been `204 No content` instead, but there were reports of some browsers (Firefox) incorrectly processing such results, so it's as it is.

Example

```
DELETE $baseUrl/rest/structure/1.0/structure/108
```

695 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

```
{
  "empty": true
}
```

Error

`403 Forbidden`	If REST API user is not logged in, does not have permissions to access Structure Plugin, or does not have Control(see page 362) access level to this structure. Response contains error entity.	`application/json`, `application/xml`
`404 Not Found`	If `id` is not an integer in `1..2^63-1`. Response entity contains a standard JIRA error HTML page.	`text/html`
`404 Not Found`	If `id` is an integer in `1..2^63-1`, but the structure with the specified `id` does not exist or the user does not have View(see page 362) access level to it.	`application/json`, `application/xml`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore(see page 485) operation may be in progress.	

Other return codes[696] are possible under the normal rules of HTTP communication.

Top(see page 897)

8.6.4.4 Forest Resource

Forest Resource is responsible for serving forests and forest updates and receiving the forest actions (change commands) from the client.

696 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

Retrieving Forest

Request

```
GET $baseUrl/rest/structure/2.0/forest/latest?s=$forestSpec
POST $baseUrl/rest/structure/2.0/forest/latest
```

Returns the hierarchical issue list (forest) of the specified structure.

Parameters:

$forestSpec	*required*	The URL-encoded JSON representation of ForestSpec[697]. See also: RestForestSpec[698].
POST content	*required*	While GET method is preferred, POST is more robust because there's no risk of exceeding URL length with large forest specifications. The content is the same JSON object (but not URL-encoded, obviously).

Example:

```
GET /rest/structure/2.0/forest/latest?s={%22structureId%22:113}
```

Retrieves latest forest for structure #113.

Response

```
{
  "spec":{"structureId":113},
  "formula":"10394:0:4/356,10332:0:14707,10374:1:5/240,10348:2:14717",
  "itemTypes":{
    "4":"com.almworks.jira.structure:type-generator",
    "5":"com.almworks.jira.structure:type-folder"
  },
  "version":{
    "signature":-1659607419,
    "version":1
  }
}
```

In this reply, the most important part is "formula", which contains serialized information about the forest.

697 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestSpec.html
698 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/rest/RestForestSpec.html

Each component (delimited by comma) represents a row and looks like this: 10374:1:5/240. 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(see page 855).

```
POST $baseUrl/rest/structure/2.0/forest/update
```

Parameters:

```
{
  "spec": { "structureId": <id> }, // use structure ID
  "version": { "signature": <signature>, "version": <version> },  // use last seen signature and version
  "actions": [
     {
         "action": "add",
         "under": 0,        // at the top level
         "after": 123,      // after row ID 123 (not issue iD!)
         "before": 456,     // before row ID 456
         "forest": "-100:0:10001"   // insert issue 10001, -100 is the temporary row ID which will be mapped into the real row ID when the method returns
     },
     {
         "action": "move", // works like previously, only row IDs instead of issue IDs
         "rowId": 123,
         "under": 456,
         "after": 0,
         "before": 124
     },
    {
       "action": "remove",
       "rowId": 442
     }
  ]
}
```

8.6.4.5 Item Resource

Item resource is used to create new items and update existing items.

Creating a New Item

The following request is used to create a new item (issue, folder or other type) and insert it into a forest.

```
POST $baseUrl/rest/structure/2.0/item/create
```

This request should upload a specification of the creation action and coordinates of where to put the result.

Example

```
{
  "item": {
    "type": "com.almworks.jira.structure:type-folder",
    "values": { "summary": "New folder name" }
  },
  "forest": {
    "spec": { "structureId": 128 },
    "version": {
      "signature": 0,
      "version": 0
    }
  },
  "items": {
    "version": {
      "signature": 0,
      "version": 0
    }
  },
  "rowId": -100,
  "under": 0,
  "after": 0,
  "before": 0,
  "parameters": {}
}
```

Parameters

Parameter (see example above)	Meaning
`item`	Defines the item being created.

Parameter (see example above)	Meaning
`item.type`	Item type (complete key of the module that provides this item's main functionality.) Use `com.almworks.jira.structure:type-folder` for folders and `com.almworks.jira.structure:type-issue` for issues. See also: CoreItemTypes[699]
`item.values`	A set of values for the new item. The specific fields depend on the item. For a folder, it is "summary". For other items, see examples below.
`forest.spec`	Forest specification of the forest that will receive the new item. See ForestSpec[700] and RestForestSpec[701].
`forest.version`	Last known version of the forest. The reply to this call will contain the update to that version. Use zero version (as in example) to receive full forest.
`items.version`	Last known version of instance items set. The reply to this call will contain an update to the known items. Use zero version (as in example) to receive full update.
`rowId`	Temporary ID assigned to the created issue. Must be negative. You can use `-100` in most cases.
`under` / `after` / `before`	Forest coordinates to insert the new item into. See Forest Resource(see page 917).

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:

699 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/CoreItemTypes.html
700 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/forest/ForestSpec.html
701 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/rest/RestForestSpec.html

```
"item": {
  "type": "com.almworks.jira.structure:type-issue",
  "values": {
    "issue": {
      "summary": "issue summary"
    },
    "pid": 10000,
    "issuetype": "3",
    "mode": "new",
  }
}
```

The above are the minimal fields needed to create a new issue. Note that `pid` is a number, but `issuetype` is a string.

Reply Example

The following is an example of a reply.

```
{
  "successfulActions": 1,
  "itemId": "com.almworks.jira.structure:type-issue/10100",
  "oldRowIds": [-100],
  "newRowIds": [61],

  "forestUpdates": [...],
  "itemsUpdate": {...}
}
```

Most important fields are `itemId` and `newRowIds`. More on the return fields:

Field	Explanation
`successfulActions`	A number of actions successfully performed by the server. In this case, it's either 0 or 1.
`itemId`	The ID of the newly created item. See ItemIdentity[702].
`oldRowIds` / `newRowIds`	Provides mapping from the temporary row IDs used for uploading the action and the real row IDs obtained after the item was inserted.
`forestUpdates`	Changes to the forest since the version passed in the request.
`itemsUpdate`	Changes to the items set since the version passed in the request.

702 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/ItemIdentity.html

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`	The ID of the item. If it is just a number, like in the example, it is an issue ID. Note that it is still a **String** value that contains issue ID. Instead of a number, it can be a canonical notation of an ItemIdentity[703]. For example, to update a folder, use `"com.almworks.jira.structure:type-folder/123"` where 123 is the folder ID.

703 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/item/ItemIdentity.html

Parameter (see example above)	Meaning
`item.values`	A map of values to be updated. The keys are the same as when the item is created. For updating a folder, use `"summary"`.
`items.version`	Known version of the items stream. The response will contain an update based on that number. Use zeroes, as in example, when updated is not needed.
`forest.spec` and `forest.version`	Monitored forest spec and known version of that forest. The response will contain a forest update based on those values. When not needed, use a simple forest (like clipboard in this example) and zeroed version.

Reply

The reply is similar to the reply from calling `/create` method, defined above. A positive HTTP status tells that the item has been updated. There is no `"itemId"` in the response.

8.6.4.6 Value Resource

Value Resource is used to retrieve values of attributes for rows in a given forest.

> To learn more about attributes, see Loading Attribute Values(see page 858).

To retrieve values from Structure, you need a few things first:

- A forest specification (`forestSpec`) for the displayed forest – same as the one used in Forest Resource(see page 917). Forest specification is needed even if the values do not depend on the forest.
- A list of row IDs for which the values should be loaded. Row IDs can be retrieved from Forest Resource before calling Value Resource.
- A list of attribute specifications. Some examples are given below.

Loading Values

To load values use the following call

```
POST $baseUrl/rest/structure/2.0/value
```

The request should come with JSON payload that specifies which values you are interested in.

Example

```
{
  "requests": [
    {
      "forestSpec": {
        "structureId": 123
      },
      "rows": [
        1820,
        1842,
        2122
      ],
      "attributes": [
        {
          "id": "summary",
          "format": "text"
        },
        {
          "id": "key",
          "format": "html"
        },
        {
          "id": "progress",
          "format": "number",
          "params": {
            "basedOn": "timetracking",
            "resolvedComplete": true,
            "weightBy": "equal"
          }
        }
      ]
    }
  ]
}
```

As you see in this example, a request body may contain one or more request, each for a specific matrix of several rows and several attributes. A value for each pair of a row and an attribute will be calculated.

Parameters

Parameter	Meaning
`requests[i].forestSpec`	Forest specification that produces the forest from which the rows are taken.
`requests[i].rows`	Array of row IDs for which values should be loaded.
`requests[i].attributes`	Array of attribute specifications that should be loaded for each row.

The example shows three attributes being loaded – plain text Summary, html-formatted Key and Progress based on time tracking. For more information about available system attributes, see javadocs for AttributeSpec[704] and CoreAttributeSpecs[705].

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.

704 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/AttributeSpec.html
705 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/CoreAttributeSpecs.html

```
{
  "responses": [
    {
      "forestSpec": {
        "structureId": 123
      },
      "rows": [
        1820,
        1842,
        2122
      ],
      "data": [
        {
          "attribute": {
            "id": "summary",
            "format": "text"
          },
          "values": [
            "Issue 1",
            "Folder 2",
            "Some Other Item 3"
          ],
          "trailMode": "INDEPENDENT",
          "trails": [ "", "", "" ]
        }
      ],
      "forestVersion": {
        "signature": -1385959428,
        "version": 1
      }
    }
  ],
  "itemTypes": {},
  "itemsVersion": {
    "signature": -558220658,
    "version": 1
  }
}
```


Parameters

Parameter	Meaning
`responses[i].forestSpec`	Requested forest spec, from which the rows are taken.
`responses[i].rows`	A list of row IDs for which the values are provided.
`responses[i].data[j].attribute`	The attribute specification for which the following values are calculated.

Parameter	Meaning
`responses[i].data[j].values`	Array of values. The value at k-th place corresponds to the row at k-th place in `responses[i].rows`.

If you are receiving value in any format other than `html`, you need to html-escape that value before adding it to the web page.

8.6.4.7 Attribute Subscription Resource

Attribute subscription resource is used to retrieve attribute values and updates to those values sequentially through a subscription.

Attribute Subscription Resource is introduced in Structure 6.0. When working with an older version, use Value Resource(see page 924).

To learn more about attributes, see Loading Attribute Values(see page 858).

Subscriptions

A subscription represents an interest of a client code in a set of attributes for a set of rows in a particular forest. Just like when loading attributes through Value Resource(see page 924), in order to create a subscription, you need:

- A forest specification (`forestSpec`) for the displayed forest.
- A list of row IDs for which the values should be loaded. Row IDs can be retrieved from Forest Resource before calling Value Resource.
- A list of attribute specifications. Some examples are given below and in the documentation for Value Resource(see page 924).

The chosen rows and attributes of interest are called a "window". The subscription server keeps track of registered subscriptions and corresponding windows, and, upon requests, calculates the updated data for a window and sends the updates back to the client.

Attribute subscription is the preferred way to continuously receive attribute values. For a one-off download, it's better to use Value Resource.

Base URL

Base URL for Attribute Subscription Resource is: `$baseUrl/rest/structure/2.0/attribute/subscription`

In the following documentation we will simply write `/attribute/subscription` meaning the full base URL above.

Common Parameters and Data Structures

There are a number of parameters that are used repeatedly in multiple methods in the Attribute Subscription Resource.

Input Query Parameters

These are passed as a part of URL. Neither of them are required.

Parameter Name	Type	Default	Meaning
`valuesUpdate`	Boolean	`false`	Indicates whether a client desires to receive the updated values for the current or updated window. This typically implies that a request to load these values will be executed.
`valuesTimeout`	Long	`1000ms`	Indicates the maximum time (in milliseconds) the server will wait for the values to be calculated before responding to the client. Applies only if `valuesUpdate` is true. The default can be adjusted through Advanced Configuration and Dark Features(see page 499).
`signature`	Integer	`0`	Together, signature and version define the last point of synchronization with the updates from the server. In the first request, the client should use zeros or not use them at all. Each successful response will include an updated signature and version, which client code should use the next time in order to get only the values that have changed.
`version`	Integer	`0`	

Data Structure: `SubscriptionWindow`

The `SubscriptionWindow` data structure represents a subscription window.

Field	Type	Meaning
`forestSpec`	Object: a forest specification, the same as in Forest Resource(see page 917)	Defines the forest for which the values will be loaded.
`rows`	Array of Long numbers	Defines a list of rows to be loaded.
`attributes`	Array of Objects: AttributeSpec	Defines a list of attributes to be loaded.

Example:

```
{
  "forestSpec": { "structureId": 1 },
  "rows": [ 10, 15, 1, 27, 1001 ],
  "attributes": [
    { "id": "summary", "format": "text" },
    { "id": "key", "format": "html" },
    {
      "id": "progress",
      "format": "number",
      "params": {
        "basedOn": "timetracking",
        "resolvedComplete": true,
        "weightBy": "equal"
      }
    }
  ]
}
```

Data Structure: `SubscriptionData`

The `SubscriptionData` data structure is sent back from most of the REST calls to the Attribute Subscription Resource.

Field	Type	Meaning
`id`	Integer	The unique ID of the subscription.
`window`	`SubscriptionWindow`	The current (updated) window for the subscription.
`valuesUpdate`	`SubscriptionUpdate`	An update for the current subscription.

Data Structure: `SubscriptionUpdate`

A subscription update is returned as a part of `SubscriptionData` structure and carries the new values, versioning information, error information and loading progress information.

This structure is also used in the Poll Resource.

Field	Type	Meaning
`id`	Integer	The unique ID of the subscription.
`full`	Boolean	If true, this is a "full" or "total" update, meaning that all old values are obsolete. If false, it is an incremental update and includes only the updated values.

Field	Type	Meaning
`fromVersion`	`RestVersion`	Contains signature and version that were used to request the update.
`version`	`RestVersion`	Contains signature and version that should be used to request the next update.
`data`	Array of Objects	Each object represents values for a particular attribute.
`data[i].attribute`	AttributeSpec	Describes the attribute for which the values are provided.
`data[i].values`	Object	A map from row IDs (represented as strings) to the attribute values. A value may take multiple forms – it could be a number, a text, a boolean, an array or an object.
`data[i].outdated`	Array of Longs	Optional field that lists the values which are known to contain outdated values. (But the values are provided all the same, so that the client can show something while the updated values are being calculated.)
`stillLoading`	Boolean	If true, not all of the requested values have been loaded. The client can retry the request some time later.
`inaccessibleRows`	Array of Longs	Optional field that lists all requested rows that are not accessible by the user (for any reason). The client code should not continue requesting them.
`error`	`StructureError`	An object containing error code and other diagnostics in case of a problem.
`attributeErrors`	A list of Objects	An optional list of attribute-specific errors.

Resource Methods

Create Subscription

```
POST /attribute/subscription
```

Query parameters	`valuesUpdate`, `valuesTimeout`	Allow immediately loading some values for the created subscription.
Input data	`SubscriptionWindow`	Passed as the request body, the window defines the subscription parameters.

Response	`SubscriptionData`	Contains the ID of the newly created subscription, the used window, and some values if they were requested.

Retrieve Subscription or Values

```
GET /attribute/subscription/ID
```

Path parameters	ID	Identifies the subscription by ID.
Query parameters	`valuesUpdate`, `valuesTimeout`	Allow immediately loading some values for the created subscription.
	`signature`, `version`	Identify the previous version of the values that the client has seen to get incremental updates.
	`skipLoading`	A Boolean parameter: if `true`, then only the already calculated values will be loaded, without requesting the attribute subsystem to reload the values in the window.
Response	`SubscriptionData`	Contains all the subscription data and value updates if they were requested.

Update Subscription

Used to completely change the subscription's window.

```
PUT /attribute/subscription/ID
```

Path parameters	ID	Identifies the subscription by ID.
Query parameters	`valuesUpdate`, `valuesTimeout`	Allow immediately loading some values for the updated subscription.
	`signature`, `version`	Identify the previous version of the values that the client has seen to get incremental updates.
Input data	`SubscriptionWindow`	The updated subscription window.
Response	`SubscriptionData`	Contains all the subscription data and value updates if they were requested.

Patch Subscription

Used to partially change the subscription's window. Only the fields mentioned in the input `SubscriptionWindow` object will be updated.

```
POST /attribute/subscription/ID/patch
```

Path parameters	ID	Identifies the subscription by ID.
Query parameters	`valuesUpdate`, `valuesTimeout`	Allow immediately loading some values for the updated subscription.
	`signature`, `version`	Identify the previous version of the values that the client has seen to get incremental updates.
Input data	`SubscriptionWindow`	The object with the parts of the subscription window that need to be updated.
Response	`SubscriptionData`	Contains all the subscription data and value updates if they were requested.

Delete Subscription

```
DELETE /attribute/subscription/ID
```

Path parameters	ID	Identifies the subscription by ID.
Response	`HTTP 200`	Simple response with no data.

User Access

When a subscription is created, it is attached to the current user. Only this user will have access to the subscription.

Subscription Expiration and Caching

A subscription is considered a transient, cache-like object. It may be removed from the server at any time, even if the user has not requested it.

When a request is made about a subscription that no longer exists, the server responds with HTTP 404 error and an object describing the error in detail. The client code may decide to re-create a subscription at that point.

8.6.4.8 Configuration Resource

This page describes resources which allow you to manage the list of projects(see page 477) for which Structure is enabled and the global permissions(see page 478) of the Structure app.

> ⓘ Only logged in Jira administrators can access this resource.

Configuration resource belongs to **version 2.0** of the API, its path is `/rest/structure/2.0/configuration`. It contains the following resources:

/projects GET(see page 934)	Get the list of projects for which Structure is enabled
/projects PUT(see page 935)	Update the list of projects for which Structure is enabled
/projects/add POST(see page 935)	Enable Structure for projects
/projects/remove POST(see page 936)	Disable Structure for projects
/permissions GET(see page 938)	Get Structure's global permissions
/permissions PUT(see page 940)	Update Structure's global permissions
/permissions/{key} GET(see page 941)	Get a particular global permission
/permissions/{key} PUT(see page 942)	Update a particular global permission
/permissions/{key}/add POST(see page 942)	Grant a global permission to user groups or project roles
/permissions/{key}/remove POST(see page 943)	Revoke a global permission from user groups and project roles

Quick navigation:

- Permission keys(see page 937)
- Permission configuration objects(see page 937)
- Permission subjects(see page 938)

Project Resources

GET /projects

Returns a JSON object which contains:

- a flag telling whether Structure is enabled for all projects, and
- the list of project IDs for which Structure is enabled.

Please note that even when Structure is enabled for all projects, the list of selected project IDs is still stored in the app configuration and returned by this resource.

```
GET $baseUrl/rest/structure/2.0/configuration/projects
```

A successful request returns a JSON object with two fields, for example:

```
{
  "enabledForAllProjects": false,
  "pickedProjectIds": [10000, 10100]
}
```

Top(see page 933)

PUT /projects

Updates the "enabled for all projects" flag and/or the list of projects for which Structure is enabled.

```
PUT $baseUrl/rest/structure/2.0/configuration/projects
```

Accepts a JSON object with the following fields, at least one of which must be present:

- `enabledForAllProjects` – a boolean indicating whether Structure is enabled for all projects. If omitted, the setting is not changed.
- `pickedProjectIds` – an array of numbers with the IDs of the projects for which Structure is enabled. If omitted, the list of projects is not changed.

Please note that when Structure is enabled for all projects, the list of projects has no effect, but it is still stored in the app configuration and can be updated.

Example 1. Enable Structure for only two projects with IDs 10000 and 10100:

```
PUT $baseUrl/rest/structure/2.0/configuration/projects


{
  "enabledForAllProjects": false,
  "pickedProjectIds": [10000, 10100]
}
```

Example 2. Enable structure for all projects, don't change the list of projects:

```
PUT $baseUrl/rest/structure/2.0/configuration/projects


{ "enabledForAllProjects": true }
```

A successful request returns an "empty" JSON object:

```
{ "empty": true }
```

Top(see page 933)

POST /projects/add

Adds one or more projects to the list of projects for which Structure is enabled.

```
POST $baseUrl/rest/structure/2.0/configuration/projects/add
```

Accepts a JSON array of numbers, all of which must be existing project IDs. At least one project ID must be supplied.

Please note that when Structure is enabled for all projects, the list of projects has no effect, but it is still stored in the app configuration and can be updated.

Example. Add project with IDs 10200 and 10300 to the list of projects:

```
POST $baseUrl/rest/structure/2.0/configuration/projects/add


[10200,10300]
```

A successful request returns a JSON object with a single boolean flag indicating whether the list of projects was updated (i.e. it did not contain at least one of the passed project IDs before the call):

```
{ "updated": true }
```

Top(see page 933)

POST /projects/remove

Removes one or more projects from the list of projects for which Structure is enabled.

```
POST $baseUrl/rest/structure/2.0/configuration/projects/remove
```

Accepts a JSON array of numbers. Non-existent project IDs are allowed. At least one project ID must be supplied.

Please note that when Structure is enabled for all projects, the list of projects has no effect, but it is still stored in the app configuration and can be updated.

Example. Remove project with ID 10200 from the list of projects:

```
POST $baseUrl/rest/structure/2.0/configuration/projects/remove


[10200]
```

A successful request returns a JSON object with a single boolean flag indicating whether the list of projects was updated (i.e. it contained at least one of the passed project IDs before the call):

```
{ "updated": true }
```

Top(see page 933)

Permission Resources

Permission Keys

The following global permissions are available:

Key	Permission
`use`	Use Structure. See Who Has Access to the Structure(see page 478).
`createStructure`	Create new structures. See Changing Permission to Create New Structures(see page 479).
`synchronization`	Configure synchronizers.
`automation`	Configure automation (generators and effectors). See Changing Permission to Access Automation(see page 480).
`configureGenerators`	Configure generators. The "automation" permission is also required.
`configureEffectors`	Configure effectors. The "automation" permission is also required. See Changing Permissions to Configure and Run Effectors(see page 480).
`executeEffectors`	Execute effectors.
`executeEffectorsOnQueries`	Execute effectors on query results.

Top(see page 933)

Permission Configuration Objects

The configuration of each global permission is represented by a JSON object with two fields:

Field	Type	Description
`allowedForAnyone`	boolean	Indicates that a permission is granted to anyone.
`subjects`	array of objects	The list of permission subjects to which the permission is granted.

The flag and the list are stored independently. If a permission is granted to anyone, the list of permission subjects has no effect, but it is still stored in the Structure app configuration and can be updated.

Also, even if a global permission is granted to anyone, there can be other permission-specific restrictions. For example, to configure generators in a structure, a user needs the "Automate" access level to that structure.

Top(see page 933)

Permission Subjects

Global permissions can be granted to user groups and project roles. Each permission subject is represented by a JSON object with the following fields:

Field	Type	Description
`subject`	string	The type of the subject. Must be either `"group"` or `"projectRole"`. Required.
`groupId`	string	Group name. Required for user groups.
`roleId`	number	Project role ID. Required for project roles.
`projectId`	number	Project ID. Required for project roles. 0 means "any project".

Example 1. The `jira-administrators` group:

```
{
  "subject": "group",
  "groupId": "jira-administrators"
}
```

Example 2. Role with ID 10002 in any project:

```
{
  "subject": "projectRole",
  "projectId": 0,
  "roleId": 10002
}
```

Top(see page 933)

GET /permissions

Returns all global permissions as a JSON object.

```
GET $baseUrl/rest/structure/2.0/configuration/permissions
```

In the returned JSON object, keys are permission keys(see page 937) and values are permission configuration objects(see page 937). Example:

```
{
  "use": {
    "allowedForAnyone": true,
    "subjects": []
  },
  "createStructure": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-software-users"
      }
    ]
  },
  "synchronization": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-administrators"
      }
    ]
  },
  "automation": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-administrators"
      },
      {
        "subject": "projectRole",
        "projectId": 0,
        "roleId": 10002
      }
    ]
  },
  "configureGenerators": {
    "allowedForAnyone": true,
    "subjects": []
  },
  "configureEffectors": {
    "allowedForAnyone": true,
    "subjects": []
  },
  "executeEffectors": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "projectRole",
        "projectId": 10000,
        "roleId": 10100
```

```
      },
      {
        "subject": "projectRole",
        "projectId": 10100,
        "roleId": 10100
      }
    ]
  },
  "executeEffectorsOnQueries": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-administrators"
      },
      {
        "subject": "projectRole",
        "projectId": 0,
        "roleId": 10002
      }
    ]
  }
}
```

Top(see page 933)

PUT /permissions

Updates one or more global permissions.

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions
```

Accepts a JSON object where keys are permission keys(see page 937) and values are permission configuration objects(see page 937). Omitted global permissions are not changed. For each permission, either field can be omitted, but at least one field must be supplied.

Example. Allow anyone to use Structure, but allow only the "jira-software-users" group to create new structures:

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions


{
  "use": {
    "allowedForAnyone": true
  },
  "createStructure": {
    "allowedForAnyone": false,
    "subjects": [
      {
        "subject": "group",
        "groupId": "jira-software-users"
      }
    ]
  }
}
```

A successful request returns an "empty" JSON object:

```
{ "empty": true }
```

Top(see page 933)

GET /permissions/{key}

Returns a permission configuration object(see page 937) for the given permission key(see page 937).

```
GET $baseUrl/rest/structure/2.0/configuration/permissions/$key
```

Example. Get the configuration object for the permission to create new structures:

```
GET $baseUrl/rest/structure/2.0/configuration/permissions/createStructure

{
  "allowedForAnyone": false,
  "subjects": [
   {
      "subject": "group",
      "groupId": "jira-software-users"
    }
  ]
}
```

Top(see page 933)

PUT /permissions/{key}

Updates the configuration of the given permission key(see page 937).

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions/$key
```

Accepts a permission configuration object(see page 937). Either field can be omitted, but at least one field must be supplied.

Example 1. Allow anyone to use Structure:

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions/use

{
  "allowedForAnyone": true
}
```

Example 2. Allow only Jira administrators and project administrators (role ID 10002) in any project to configure automation:

```
PUT $baseUrl/rest/structure/2.0/configuration/permissions/automation

{
  "allowedForAnyone": false,
  "subjects": [
    {
      "subject": "group",
      "groupId": "jira-administrators"
    },
    {
      "subject": "projectRole",
      "projectId": 0,
      "roleId": 10002
    }
  ]
}
```

A successful request returns an "empty" JSON object:

```
{ "empty": true }
```

Top(see page 933)

POST /permissions/{key}/add

Adds user groups and/or project roles to the list of permission subjects for the given permission key(see page 937).

```
POST $baseUrl/rest/structure/2.0/configuration/permissions/$key/add
```

Accepts a JSON array of one or more permission subjects(see page 938). If the permission is granted to anyone, the list of permission subjects has no effect, but it is still stored in the Structure app configuration and can be updated.

Example. Grant project administrators (role ID 10002) in any project the permission to execute effectors on query results:

```
POST $baseUrl/rest/structure/2.0/configuration/permissions/executeEffectorsOnQueries/
add

[
  {
    "subject": "projectRole",
    "projectId": 0,
    "roleId": 10002
  }
]
```

A successful request returns a JSON object with a single boolean flag indicating whether the list of permission subjects was updated (i.e. it did not contain at least one of the passed subjects before the call):

```
{ "updated": true }
```

Top(see page 933)

POST /permissions/{key}/remove

Removes user groups and/or project roles from the list of permission subjects for the given permission key(see page 937).

```
POST $baseUrl/rest/structure/2.0/configuration/permissions/$key/remove
```

Accepts a JSON array of one or more permission subjects(see page 938). If the permission is granted to anyone, the list of permission subjects has no effect, but it is still stored in the Structure app configuration and can be updated.

Example. Revoke the global permission to configure synchronizers from Jira Software users:

```
POST $baseUrl/rest/structure/2.0/configuration/permissions/synchronization/remove

[
  {
    "subject": "group",
    "groupId": "jira-software-users"
  }
]
```

A successful request returns a JSON object with a single boolean flag indicating whether the list of permission subjects was updated (i.e. it contained at least one of the passed subjects before the call):

```
{ "updated": true }
```

Top(see page 933)

8.6.5 Structure JavaScript API Reference

Structure's JavaScript API provides ways to extend the client-side functionality of the Structure plugin.

- JavaScript API Functions(see page 944) — This page lists static functions exposed by the Structure API.
- JavaScript API Classes(see page 946) — Structure Javascript API provides a number of classes to be used as base for your own column type implementations. This should be done using subClass() method.
    - Column Class(see page 947) — A subclass of Column class represents column objects of a specific type.
    - ColumnConfigurator Class(see page 951) — ColumnConfigurator class encapculates everything related to column type configuration.
    - ColumnOption Class(see page 953) — ColumnOption class represents a single column configuration parameter.
    - ColumnType Class(see page 956) — ColumnType class represents column type.
    - ItemDetailsProvider Class(see page 958) — ItemDetailsProvider class is the extension point for item details.
- JavaScript API Objects(see page 962) — This page lists objects exposed by the Structure API.
    - ItemDetailsBridge Object(see page 962) — ItemDetailsBridge provides api for interaction with the Structure app and item details lifecycle.

8.6.5.1 JavaScript API Functions

This page lists static functions exposed by the Structure API.

window.almworks.structure.api.subClass(className, superclass, prototype)

Creates a subclass of a specific class. Returns a constructor function that will create the instances of the class.

This function provides light-weight polymorphism for the purposes of extending Structure's classes(see page 946).

Parameters

`className`	*string*	Class name as string (optional, used for friendly instance names in debugger)
`superclass`	*Object*	Superclass reference
`prototype`	*Object*	Subclass prototype

The returned value – class constructor – takes a single optional `options` parameter.

The prototype may contain a special `init()` initializer method, which is called when an instance is being constructed. Superclass' `init()` method is called before subclass' method. Options that were passed to the constructor are passed through to the initializer.

Example

```
var MyClass = window.almworks.structure.api.subClass('MyClass', BaseClass, {
  init: function(options) {
   ...
  },

  someMethod: function() {
   ...
  }
});

var options = { ... };
var instance = new MyClass(options);
```

window.almworks.structure.api.registerColumnType(type, key)

Registers a new column type. If you're extending Structure by adding a new type of column to the grid, the type must be registered from your additional JavaScript web resource.

Column types are identified by a unique key, which is recorded in the view(see page 341) specification, along with the type-specific parameters and column name.

Parameters

`type`	*Object*	A `ColumnType` instance, implementing a specific column type – see ColumnType Class(see page 956)
`key`	*string*	Column type key (can also be array of strings if the type can handle multiple variations of a column specification)

Example

```
window.almworks.structure.api.registerColumnType(new MyColumnType(),
'com.acme.structure.awesome-column');
```

We recommend using a unique key that has low chance of conflicting with column types provided by other, independent developers. A good approach is to have Java-like package notation for the keys.

window.almworks.structure.api.registerColumnGroup(options)

Registers a new column type group. Column groups are used in the "Add Column" panel to group column configuration presets, provided by column types.

Parameters

`options.groupKey`	*string*	Group key. The same key should be returned by all `ColumnConfigurator`'s `getGroupKey()` method for all column types which need to appear in this particular group.
`options.title`	*string*	Group title
`options.order`	*number*	Group order is used to sort groups. Order value for groups provided by Structure are 100, 200, 300 and so on.

Example

```
window.almworks.structure.api.registerColumnGroup({
  groupKey: 'com.acme.structure.colgroup', title: 'Acme Columns', order: 50
});
```

window.almworks.structure.api.registerItemDetailsProvider(itemType, ProviderClass)

Registers item details support for the given item type.

Parameters

`itemType`	*string*	Item type for which details are registered
`ProviderClass`	*Object*	Subclass of ItemDetailsProvider[706] 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);
```

8.6.5.2 JavaScript API Classes

Structure Javascript API provides a number of classes to be used as base for your own column type implementations. This should be done using subClass()(see page 0) method.

706 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class

- Column Class(see page 947) — A subclass of Column class represents column objects of a specific type.
- ColumnConfigurator Class(see page 951) — ColumnConfigurator class encapculates everything related to column type configuration.
- ColumnOption Class(see page 953) — ColumnOption class represents a single column configuration parameter.
- ColumnType Class(see page 956) — ColumnType class represents column type.
- ItemDetailsProvider Class(see page 958) — ItemDetailsProvider class is the extension point for item details.

Column Class

window.almworks.structure.api.Column

A subclass of Column class represents column objects of a specific type.Columns need to be subclassed for a particular column type implementation. You can override methods while subclassing to modify the default behavior.

Example

```
var api = window.almworks.structure.api;
var MyColumn = api.subClass('MyColumn', api.Column, {
  init: function() {
    ...
  },
  getCellViewHtml: function() {
    return '<div> ... </div>';
  }
});
```

Properties

`context`
Contains context information about where the column is used. See The Column Context(see page 869) for more information.

`spec`
Contains column specification object. Specification object is serialized as a part of the overall view specification and stored on the server and in the browser's local storage. See Column Specifications(see page 868) for more information.

Methods

`init(options)`
Initializer method.

`getCellValueHtml(renderingParameters)`
Returns HTML that is displayed in the grid cell for a specific issue. The HTML should contain the value provided by this column. Structure will also wrap the value in decorative elements – this could be overridden by providing `getCellViewHtml()` method.

Parameters

renderingParameters.getAttributeValue()	returns current row's attribute value

renderingParameters.getRowId()	returns current row's id
renderingParameters.getItemId()	return current row's item id

Example

```
var Template = require('almworks/util/Template');
var cellTemplate = new Template('<span class="acme-field">{awesomefield}</span>');
getCellValueHtml: function(rp) {
  return cellTemplate.renderHtml({ awesomefield: rp.getAttributeFieldValue({id:
'com.acme.awesome-data', format: 'text'}) });
}
```

`getCellViewHtml(renderingParameters)`
Returns customized HTML that is displayed in the grid cell for a specific issue. By default, calls `getCellValueHtml()` and wraps the retrieved value into the default Structure style. Can be overridden to allow higher degree of control over the cell appearance.

Parameters

renderingParameters.getAttributeValue()	returns current row's attribute value
renderingParameters.getRowId()	returns current row's id
renderingParameters.getItemId()	return current row's item id

`collectRequiredAttributes(attributeSet)`
Lets column request attributes that are needed for rendering. The attributes are provided on the server side by AttributeLoaderProvider[707].

Parameters

attributeSet.requireAttribute(attributeSpec,forestSpec)	Method for collecting required attributes. Parameters are: • `attributeSpec` is the attribute specification object • `forestSpec` is the forest specification for the forest, from which attribute should be loaded **(optional)**

About Attribute Specs

AttributeSpec defines the attribute and format to be loaded. See Loading Attribute Values(see page 858) 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.

707 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/loader/AttributeLoaderProvider.html

About Forest Spec

Forest specification is optional. When used, it allows you to get attribute value from a different forest – however, it must be related to the forest being displayed, otherwise it will not have the same rows.

For example, you can specify a forest specification with some transformation to display values from there in the untransformed forest. There are also two special values for `forestSpec`:

- `'displayed'` is the default value, meaning "use the forest that is being displayed"
- `'unfiltered'` means "use the same forest, but remove all filters that are coming at the end of transformation chain"

Example

```
collectRequiredAttributes: function(attributeSet) {
  attributeSet.requireAttribute({id: 'key', format: 'text'});
  attributeSet.requireAttribute({
    id: 'sum',
    format: 'number',
    params: {
      id: 'customfield',
      format: 'number',
      params: {
        fieldId: 10010
      }
    }
  }, 'unfiltered');
  attributeSet.requireAttribute({id: 'com.mycompany.work-stats', format: 'json'});
}
```

Some of the attributes provided by Structure:

Attribute Spec	Example	Description
`{id: <jira-field-id>, format: 'html'}`		The HTML representation of a JIRA issue field value, as seen on the issue page or in the Issue Navigator. Structure allows non-issue items also have these values. `<jira-field-id>` is the common name for the JIRA's standard field id. This attribute does not load custom fields.
`{id: 'customfield', format: 'html', params: { fieldId: <field-numeric-id> }}`		HTML representation of a custom field value.

Attribute Spec	Example	Description
`{id: 'project', format: 'id'}`		Project ID for the issues. The `id` format means either a string or a number, depending on what is being used for identifying the object.
`{id: 'editable', format: 'boolean'}`		Boolean value telling whether the item can be edited by the user.

See also CoreAttributeSpecs[708] 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()`

708 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/attribute/CoreAttributeSpecs.html

Returns minimum width of the column in pixels. Returns 27 by default.

Example

```
getMinWidth: function() { return 100; }
```

`getDefaultWidth()`
Returns default width of the column in pixels. Returns 120 by default.

Example

```
getDefaultWidth: function() { return 100; }
```

`getHeaderCellHtml()`
Returns HTML that will be used in the grid header. By default returns cell with column name in default Structure style.

Example

```
getHeaderCellHtml: function() { return '<div>' + this.name + '</div>'; }
```

`getMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column to render the values. See Requesting and Using Metadata(see page 871) for more information. By default returns `null`, which means that no metadata is needed.

Example

```
getMetadataRequests: function() {
  return {
    status: {
      url: baseUrl + '/rest/api/2/status',
      cacheable: true
    }
  };
}
```

`getSortAttribute()`
Returns attribute specification for sorting when the user clicks on the header. If `null` is returned (the default), the clicking this column header does not result in added sorting transformation.

`isSortDescendingByDefault()`
If returns `true` the initial direction of the sorting will be descending.

ColumnConfigurator Class

window.almworks.structure.api.ColumnConfigurator

ColumnConfigurator class encapculates everything related to column type configuration.

It needs to be subclassed for a particular column type implementation and passed as return value in ColumnType.createConfigurator()(see page 957) method.

Example

```
var api = window.almworks.structure.api;

var MyColumnConfigurator = api.subClass('MyColumnConfigurator,
api.ColumnConfigurator, {
  getDefaultColumnName: function() { return 'My Column'; }
  getOptions: function() {
    return [ new MyOption1({configurator: this}), new MyOption2({configurator: this})
 ];
  }
});
```

Required Methods

You have to override the following methods in the subclass.

`getColumnTypeName()`
Returns column type name, used in the column configuration panel.

`getDefaultColumnName()`
Returns default column name.

Other Methods

These methods may be optionally overridden.

`init(options)`
Optional initializer.

`getGroupKey()`
Return column preset's group key. See registerColumnGroup()(see page 945) for reference.

`getMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this configurator to set up the UI. See Requesting and Using Metadata(see page 871) for more information. By default returns `null`, which means that no metadata is needed.

Example

```
getMetadataRequests: function() {
  return {
    somedata: {                         // metadata key
      url: baseUrl + '/some/data/url', // request URL
      cacheable: true,                  // if the response for this URL can be reused
for other cacheable requests
      extract: function(response) {    // response to the AJAX request
        return response.property || 1; // the actual value for
context.getMetadata('somedata')
      }
    },
    otherdata: {
      url: baseUrl + '/other/data/url',
      cacheable: true
    }
  };
}
```

`getOptions()`
Returns array of column type options. Each option should be a subclass of ColumnOption Class(see page 953).

Example

```
getOptions: function() {
  return [ new MyOption1({configurator: this}), new MyOption2({configurator: this}) ]
;
}
```

ColumnOption Class

window.almworks.structure.api.ColumnOption

ColumnOption class represents a single column configuration parameter.

It needs to be subclassed for particular column type implementation and passed as return value in ColumnConfigurator.getOptions()(see page 953) method.

Options are displayed in column configuration dialog one after another with labels on the left and inputs on the right.

Example

```
var api = window.almworks.structure.api;

var MyOption1 = api.subClass('MyOption', api.ColumnOption, {
  title: 'Some option',
  init: function() {
    this.input$ = null;
  },
  createInput: function(div$) {
    this.input$ = div$.append('<input type="text" class="text">').find('input');
    var params = this.spec.params;
    this.input$.on('change', function() {
      if (params.someOptionAvaiable) {
        params.someOption = $(this).val();
        div$.trigger('notify');
      }
    });
  },
  notify: function() {
    var available = this.spec.params.someOptionAvaiable;
    this.input$.val(available ? (this.spec.params.someOption || '42') : '');
    return available;
  }
});
```

Properties

`title`
If set, title is displayed as a label to the left of the input controls. Option title representation may be overridden in #createLabel(div$)(see page 955) method.

Required Methods

You need to override the following methods.

`createInput(div$)`
Should be overridden to provide custom HTML for the option input. `div$` parameter provides parent option element to append your view to. Created input should trigger `'notify'` event on `div$` to notify Structure of any column parameters change.

Please honor the AUI Forms HTML layout when creating your input controls!

Example

```
createInput: function(div$) {
  var self = this;
  this.input$ = $('<input type="text" class="text">').appendTo(div$).on('change',
function() {
    if (self.spec.params.myOption !== $(this).val()) {
      self.spec.params.myOption = $(this).val();
      div$.trigger('notify');
    }
  });
}
```

Other Methods

`init(options)`
Optional initializer.

`createLabel(div$)`
May be overridden to provide custom HTML view for the input label. `div$` parameter provides parent option element to append your view to. By default creates a right-aligned label with text of the #title(see page 954) property.

Please honor the AUI Forms HTML layout if you override this method!

`notify()`
This method is called when the column configuration has changed. The implementation may want to update its controls to reflect those changes. The method should return a `boolean` indicating whether this option is available. Unavailable options will not be shown on the configuration panel. The default implementation does nothing and always returns `true`.

Example

```
notify: function() {
  this.input$.val(this.spec.params.myOption);
  return true;
}
```

`isInputValid()`
Returns `true` if the current column specification is valid from the point of view of this option. The column configuration won't be saved unless all of the options approve the specification. The default implementation does nothing and returns `true`.

Example

```
isInputValid: function() {
  // Check that the "field" specification parameter is present.
  return !!this.spec.params.field;
}
```

ColumnType Class

window.almworks.structure.api.ColumnType

ColumnType class represents column type.

It needs to be subclassed for particular column type implementation.

Example

```
var api = window.almworks.structure.api;

var AwesomeColumnType = api.subClass('AwesomeColumnType', api.ColumnType, {
  createSwitchTypePreset: function(context) { return { key:
'com.acme.structure.awesome-column', params: {} }; },
  createAddColumnPresets: function(context) { return [
   { key: 'com.acme.structure.awesome-column', params: {} },
   { key: 'com.acme.structure.awesome-column', name: 'Awesome Column with a Twist',
params: { twist: true } }
  ]; },
  createConfigurator: function(context, spec) { return new
 AwesomeColumnConfigurator({context: context, spec: spec}); },
  createColumn: function(context, spec) { return new AwesomeColumn({context: context,
spec: spec}); }
});

api.registerColumnType(new AwesomeColumnType(), 'com.acme.structure.awesome-column');
```

Methods

`createSwitchTypePreset(context)`
Returns default column specification to use when the user switches to this column type from another column type in the column configuration panel. May return `null` if the column type is unavailable.

`createAddColumnPresets(context)`

Returns an array of column presets (specifications) for this type to be offered to the user in the Add Column panel. May return an empty array if the column type is unavailable.

`createColumn(context, spec)`
Returns a new instance of `Column` subclass for the specified column specification. May return `null` if the specification is invalid, the column type is unavailable, etc.

`createConfigurator(context, spec)`
Returns a new instance of `ColumnConfigurator` subclass for the specified column specification. May return `null` if the specification is invalid, the column type is unavailable, etc.

`getPresetMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column type to create presets. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createSwitchTypePreset()` or `createAddColumnPresets()` is called. See Requesting and Using Metadata(see page 871) for more information. By default returns `null`, which means that no metadata is needed.

`getColumnMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column type to create `Column` instances. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createColumn()` is called, and also will be available to the created `Column` instance via `this.context`. See Requesting and Using Metadata(see page 871) for more information. By default returns `null`, which means that no metadata is needed.

`getConfigMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column type to create `ColumnConfigurator` instances. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createConfigurator()` is called, and also will be available to the created `ColumnConfigurator` instance via `this.context`. See Requesting and Using Metadata(see page 871) for more information. By default returns `null`, which means that no metadata is needed.

`getMetadataRequests()`
Returns a JavaScript object specifying the metadata needed by this column type. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createSwitchTypePreset()`, `createAddColumnPresets()`, `createColumn()`, or `createConfigurator()` is called, and also will be available to the created `Column` and `ColumnConfigurator` instances via `this.context`. See Requesting and Using Metadata(see page 871) 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()(see page 944) function, define necessary properties and then register provider subclass for the specific item type using registerItemDetailsProvider()[709].

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);
```

709 https://wiki.almworks.com/display/structure2gmaster/JavaScript+API+Functions#JavaScriptAPIFunctions-window.almworks.structure.api.registerItemDetailsProvider(itemType,ProviderClass)

Each ItemDetailsProvider subclass has access to ItemDetailsBridge[710] 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()[711]. 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()[712] 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[713] 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`)
  .

710 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsBridge+Object
711 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
712 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-refreshDetails(rowData,life)
713 https://api.jquery.com/category/deferred-object/

If the returned promise doesn't resolve or reject in a timely manner, Structure rejects it with argument equal to provider property timeoutMessageKey[714]. If another row is focused before this promise is done, Structure rejects it automatically.

Default implementation returns a resolved deferred object with `rowData`.

Parameters

`rowData`	*Object*	Holds current row data: • `'rowId'`: *Number* • `'itemId'`: *String*

`showDetails(rowData, life)`
Must be implemented in subclass. The main method - displays the details for the current row in the viewport.

Returns jQuery.Deferred[715] that must resolve when the details are fully shown and the user can interact with them, or reject if the details cannot be shown, with the following parameters:

- `'reason'`: *String* - is used as a i18n key in `'s.itemDetails.stub.title.+$reason+'` and `'s.itemDetails.stub.body.+$reason+'`,
- `'isError'`: *Boolean* - if details should display reason message decorated as error (optional, default: `false`)
.

If another row is focused while details are being loaded, Structure rejects it.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[716]
`life`	*Object*	Lifespan for showing details - when it finishes, we no longer need to show the details. Function that may cancel details loading or do some cleanup on lifespan finish can be registered via `life.addDetach(function)`

`beforeHide(rowData)`
Called just before the panel is hidden to perform necessary cleanup.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[717]

714 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-timeoutMessageKey
715 https://api.jquery.com/category/deferred-object/
716 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
717 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)

`hasUnfinishedEdits()`
Called before hiding details or switching it to another item to check if details has unfinished edits. Confirm dialog appears before details hiding if method returns `true`.

Should be used to prevent unexpected details hiding during content editing.

`getHeaderTitle(rowData)`
Returns text (i18nized) to set to the panel title in the details panel header.

If this method is not overridden or returns a falsy value, a title value is set to `'Item Details'`.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[718]

`getIcon(rowData)`
Returns icon to be displayed in the details header. Return value can be either html string or promise that must be resolved with a html string if it requires asynchronous loading.

The default implementation loads icon html attribute for the focused row.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[719]

`applyWidth(width)`
Updates the displayed details content so it fits the given width. Called on changing details panel width.

Parameters

`width`	*Number*	New panel width

`onDetailsFocusing()`
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[720] is set to `true`.

718 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
719 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
720 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-refreshOnStructureUpdate

Returns a jQuery.Deferred[721] that must resolve when item details has been refreshed.

Parameters

`rowData`	*Object*	Row data as returned by extendFocusedRowData()[722]
`life`	*Object*	Lifespan for showing details - when it finishes, we no longer need to show details. Function that may cancel details loading or do some cleanup on lifespan finish can be registered via `life.addDetach(function)`

`setDebug(debug)`
Controls additional debug information in logs.

Parameters

`debug`	*Boolean*	Enable/disable debug logging

8.6.5.3 JavaScript API Objects

This page lists objects exposed by the Structure API.

- ItemDetailsBridge Object(see page 962) — ItemDetailsBridge provides api for interaction with the Structure app and item details lifecycle.

ItemDetailsBridge Object

ItemDetailsBridge provides api for interaction with the Structure app and item details lifecycle.

Item details lifecycle: enabled (panel was open) → visible (ready for interaction) → disabled (panel was closed) → detached (ItemDetails module deactivated).

ItemDetailsBridge instance is accessible in ItemDetailsProvider[723] subclasses via `'itemDetailsBridge'` property.

Properties

`panel$`
jQuery object that contains item details panel element. All item details viewports[724] are appended to this container.

Methods

`getCurrentRowData()`

721 https://api.jquery.com/category/deferred-object/
722 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
723 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class
724 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-viewport

Returns `rowData` as returned by corresponding ItemDetailsProvider.extendFocusedRowData()[725] if the current row has the same item type as for which provider is registered, or `null` if it has another type.

`onCurrentRowChange(listener)`
Bind listener to execute each time on changing focused row in Structure.

Parameters

`listener`	*Function(Object)*	function that accepts `rowData` as returned by ItemDetailsProvider.extendFocusedRowData()[726]

`toggleEnabled(enabled)`
Enable/disable details panel layout.

Parameters

`enabled`	*Boolean*	`true` if panel should be enabled and `false` otherwise

`onEnableToggling(listener)`
Bind listener to execute each time the panel layout has been enabled or disabled.

Parameters

`listener`	*Function(Boolean)*	function that accepts `enabled` flag

`isEnabled()`
Returns `true` if panel layout is enabled and `false` otherwise.

`whenVisible(listener)`
Bind listener to execute when details panel becomes ready for interaction.

Parameters

`listener`	*Function*	function to execute

`isVisible()`
Returns `true` if user can interact with the details panel at the moment and `false` otherwise.

`onDetachment(listener)`
Bind listener to perform cleanup on `ItemDetails` module deactivation.

Parameters

`listener`	*Function*	function that performs cleanup

`notifyDetailsEditStarted()`
Should be called when user starts item details content editing. This prevents details refreshing after possible structure updates.

`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()`.

---

725 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)
726 https://wiki.almworks.com/display/structure2gmaster/ItemDetailsProvider+Class#ItemDetailsProviderClass-extendFocusedRowData(rowData)

`notifyDetailsUpdated()`
Should be called when user has finished details update. Needed to notify Structure about possible updates to refresh corresponding rows.

`notifyFocusChanged(focused)`
Should be called when details panel has been focused or lost its focus.

Parameters

`enabled`	*Boolean*	`true` if the panel got focus and `false` otherwise

`shouldBlockActions()`
Returns `true` if item details actions should be blocked at the moment.

`requestRefreshDetails(deferred)`
Request item details refresh. Details panel will be ready for interaction after refresh when corresponding row is updated in Structure. Can be used for Structure and details updates synchronization.

Parameters

`deferred`	*Object*	jQuery.Deferred[727] that must resolve when details panel has been refreshed and Structure row has been updated

`isFocused()`
Returns `true` if details panel is currently focused.

`setFocused(focused)`
Called to make details panel focused/unfocused.

Parameters

`focused`	*Boolean*	`true` if the panel should be focused and `false` otherwise

`focusStealerStarts(stealerId, isFocusedNow)`
Details panel focus can be temporarily taken by some UI elements or dialogs and should be returned back after its closing (or it can be some other circumstances). To provide focus retrieval this function should be called when stealer takes focus from details panel. `focusStealerFinishes()` should be called with the same `stealerId` to return focus back.

Parameters

`stealerId`	*String*	identifier of entity that stole focus from details panel
`focused`	*Boolean*	indicates that details panel should be considered as focused even if it is not so at the moment of function call

`focusStealerFinishes(stealerId, preventFocusing)`
This function should be called when UI element or dialog that stole focus should return it back (i.e. dialog has been closed or disappeared). See `focusStealerStarts()`. Details panel will be focused back when all stealers release their focus.

Parameters

`stealerId`	*String*	identifier of entity that released focus

727 https://api.jquery.com/category/deferred-object/

`preventFocusing`	*Boolean*	indicates that details panel should not be focused even if all focus stealers were released

8.6.6 Web Resource Contexts

The resources from the following web resource contexts are included by Structure pages:

Web resource context	Pages that include it
`structure.widget`	Structure Board(see page 83), Structure Gadget(see page 394) See Loading Additional Web Resources For Structure Widget(see page 879) for the recommended way of extending the widget.
`structure.printable`	Printable page(see page 380)

For details about how to use web resource contexts, see Atlassian Developer Documentation[728].

8.7 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.

8.7.1 Download

File	Modified
user-item-details-1.0.1.jar[729]	Jun 03, 2020 by Julia Atlygina[730]
status-bar-column-3.0.0.jar[731]	Jun 03, 2020 by Julia Atlygina[732]
labels-extender-1.0.3.jar[733]	Jun 03, 2020 by Julia Atlygina[734]
custom-itemtype-2.0.0.jar[735]	Jun 03, 2020 by Julia Atlygina[736]

728 https://developer.atlassian.com/display/JIRADEV/Web+Resource+Plugin+Module#WebResourcePluginModule-WebResourceContexts
729 https://wiki.almworks.com/download/attachments/32223506/user-item-details-1.0.1.jar?api=v2
730 https://wiki.almworks.com/display/~atlygina
731 https://wiki.almworks.com/download/attachments/32223506/status-bar-column-3.0.0.jar?api=v2
732 https://wiki.almworks.com/display/~atlygina
733 https://wiki.almworks.com/download/attachments/32223506/labels-extender-1.0.3.jar?api=v2
734 https://wiki.almworks.com/display/~atlygina
735 https://wiki.almworks.com/download/attachments/32223506/custom-itemtype-2.0.0.jar?api=v2
736 https://wiki.almworks.com/display/~atlygina

File	Modified
structure-api-examples-6.0.0.zip[737]	Jun 03, 2020 by Julia Atlygina[738]

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.

8.7.2 Example List

Sample Plugin	Description
simple-plugin	Very basic demo of using `StructureManager`.
status-bar-column	Adds a column to the Structure widget that shows a colored bar, depending on the statuses of the sub-issues.
labels-extender	A plugin that adds Labels Extender, which includes issues in the structure based on issue key mentioned in Labels field of the parent issue.
custom-itemtype	A plugin that adds a new item type based on Jira projects and an inserter which adds projects from one or more categories.
user-item-details	A plugin that adds item details panel support for Jira users.

8.8 Archived Documentation

- Structure 3 API Changes(see page 967)
- Creating a New Synchronizer(see page 973)
- structure-synchronizer(see page 974)

737 https://wiki.almworks.com/download/attachments/32223506/structure-api-examples-6.0.0.zip?api=v2
738 https://wiki.almworks.com/display/~atlygina

8.8.1 Structure 3 API Changes

8.8.1.1 State of the API

In Structure 3 we had to change API in an incompatible way because the underlying architecture of the product had changed. If you have integration with Structure 2, most likely it won't work with the new Structure and some effort is needed to migrate the code.

As of Structure versions 3.0 – 3.1, the new API is not yet finalized and thorough documentation is not yet published. We plan to spend additional effort on making the APIs simple, stable and well-documented and publish the final documentation then.

Until that time, it's possible to use the current non-published API with Structure 3, however:

- There's no public documentation on it. The sources of the API artifact are published, but they mostly don't have javadocs yet.
- There will be backwards-incompatible changes while we finalize the API. The concepts and interfaces will stay mostly the same, but some classes may be moved and optimized. This is less likely to impact REST API, although we plan to introduce new REST APIs that would be simpler than the low-level API we have now.
- There will be new interfaces that would make it easier to deal with the new concepts. Right now it may be a little "low-level" and somewhat more complicated than it needs to be.

Although the documentation about the new API is not available, Structure team will be happy to assist you in migrating your code to work with Structure 3.

This article lists some of the most frequently used API calls. If you need to do something that is not covered by this article or have any questions, please write us at support@almworks.com[739]

8.8.1.2 Conceptual Changes

Forests and Rows

In Structure 2, a structure's content was called a ***forest*** . That is still the case, however, the forest now contains ***rows*** rather than issues. A row has a Row ID – a long integer primary key for the row. Given row ID, you can retrieve information about the item displayed in that row.

The data structure that represents a forest didn't change. Previously a forest was represented by an array of pairs `(issueID, depth)`. Now the forest is represented by an array of pairs `(rowID, depth)`.

The concept of a row may seem superfluous, but it's actually required for uniquely identifying a specific position in a forest. Issue ID (or Item ID) is not sufficient because an issue can be located at multiple places in the forest.

Items

Each row has an associated ***item*** – an issue, a folder, a project or any other type of items. In Structure 3, item types are extendable and an add-on may provide additional types of items to Structure. An item is identified by ***Item Identity*** , which consists of:

739 mailto:support@almworks.com

- Item Type, represented by a complete module key of a JIRA plugin module that provides the item type, and
- Item ID, represented by *either* a long integer (for example, issue ID for issues) or by a string (a user key for users).

Sometimes item type is omitted; in that case the implied item type is "issue".

Some of the most popular item types are:

Item Type	Module Key	Meaning of Item ID	Comments
Issue	`com.almworks.jira.structure:type-issue`	Issue ID (long)	Default when item type is not specified
Folder	`com.almworks.jira.structure:type-folder`	Either folder ID (long) or folder i18n name (string)	Folders are introduced in Structure 3
User	`com.almworks.jira.structure:type-user`	User key (string)	
Generator	`com.almworks.jira.structure:type-generator`	Generator ID (long)	A generator is an automation rule embedded in the structure.
Page	`com.almworks.structure.pages:type-confluence-page`	Page ID (ID modulo 1e9)	Confluence pages are added as a type by Structure.Pages extension

Attributes

Attributes are a generalization of a JIRA's issue fields. An attribute is something that can be calculated for an item. For example, an issue has such attributes as "summary", "key", "priority". But purely Structure-related values are also attributes, such as "sequential number", or "aggregate progress", or "sum of story points". The attributes can also be retrieved for any types of items – for a Confluence page (provided by Structure.Pages extension), "summary" would be the title of the page, "labels" would be the labels, and a new attribute "author" would provide the initial author of the page.

Attributes are identified by an attribute specification, or ***attribute* spec**. It is usually represented as a JSON object with an ID and parameters.

Concept Comparison

	Structure 2	Structure 3
A structure's content is ...	Forest	Forest
Things that can be placed into a structure are ...	Issues	Items (including issues)
Forest consists of ...	Issues	Rows
A position in a forest is identified by ...	Issue ID	Row ID

	Structure 2	Structure 3
A value in the Structure grid is displayed by ...	Column	Column
A column requests from the server ...	Fields	Attributes

8.8.1.3 REST API

Retrieving Structure Forest

```
GET /rest/structure/2.0/forest/latest?s={%22structureId%22:$id}
```

This method retrieves a content of a structure. If structure has generators, the generated content is returned. Generators are preserved in the forest.

Parameters:

- `$id` – structure ID

Return value sample:

```
{
  "spec":{"structureId":171},
  "formula":"10394:0:4/356,10332:0:14707,10374:1:5/240,10348:2:14717",
  "itemTypes":{
    "4":"com.almworks.jira.structure:type-generator",
    "5":"com.almworks.jira.structure:type-folder"
  },
  "version":{
    "signature":-1659607419,
    "version":1
  }
}
```

In this reply, the most important part is "formula", which contains serialized information about the forest, much like in Structure 2.

Each component (delimited by comma) represents a row and looks like this: `10374:1:5/240`. In this example, the numbers are:

- `10374` is the row ID,
- `1` is the row depth,
- `5/240` is the item identity. If the row contains an issue, it's just issue ID, otherwise it has the format of `<item type>/<long item id>`, or `<item type>//<string item id>`. Item type is a number, which is expanded in the "itemTypes" map in the reply.

Updating a Structure Forest

```
POST /rest/structure/2.0/forest/update
```

Parameters:

```
{
  "spec": { "structureId": <id> }, // use structure ID
  "version": { "signature": <signature>, "version": <version> },  // use last seen signature and version
  "actions": [
     {
         "action": "add",
         "under": 0,        // at the top level
         "after": 123,      // after row ID 123 (not issue iD!)
         "before": 456,   // before row ID 456
         "forest": "-100:0:10001"   // insert issue 10001, -100 is the temporary row ID which will be mapped into the real row ID when the method returns
     },
     {
         "action": "move", // works like previously, only row IDs instead of issue IDs
         "rowId": 123,
         "under": 456,
         "after": 0,
         "before": 124
     },
    {
       "action": "remove",
       "rowId": 442
     }
  ]
}
```

Creating a structure

```
POST /rest/plugins/structure/2.0/structure
```

Parameters:

```
{
   "name": "my structure",
   "description": "my description",
   "permissions": [   ]    // same format you see when you GET structure
}
```

Deleting a structure

```
DELETE /rest/plugins/structure/2.0/structure/<id>
```

8.8.1.4 Java API

Versions

As Structure 3 API is finalized, it's getting a lot of refactoring and version changes. A new Structure version may have a backward-incompatible API, although incompatibilities may be isolated and your code has a good chance to work fine. However, the major version is promoted every time a backward-incompatible change is made, therefore you need to carefully set up the version of imported API packages – either set them optimistically (for example, `[12,15)` – up to version 15) and test your integration with a new release to see that there are no errors; or set the version as usual – for example, `[12,13)` – but then you might need to recompile with each new release of the API. The latter approach is recommended for in-house customizations.

Version	Supported JIRA Versions	Introduced in Structure Version	OSGi Import	OSGI Import (Optimistic)
12.0.0	JIRA 6.3+	3.0.0	"[12,13)"	"[12,15)"
12.1.0	JIRA 6.3+	3.0.1	"[12.1,13)"	"[12.1,15)"
13.0.0	JIRA 6.3+	3.1.0	"[13,14)"	"[13,16)"
13.0.1	JIRA 6.3+	3.1.1	"[13,14)"	" [13,16)"

The API versions and sources are available from the public Maven repositories – http://mvnrepository.com/artifact/com.almworks.jira.structure/structure-api

Retrieving Structure's Forest

To get the content of a structure, you need to use `ForestService` interface, which can be injected. It has `getForestSource()` method that will return a `ForestSource` given `ForestSpec`, which is a specification of what kind of forest you are retrieving. For getting just a content of a structure, use `ForestSpec.structure(structureId)`. Once you have a `ForestSource`, you can use `forestSource.getLatest().getForest()` to retrieve an instance of `Forest` – which should be familiar from the Structure 2 API.

But now `Forest` contains row IDs, not issue IDs, so to get information about what issues (or other items) are in the forest, you need to "dereference" each row ID.

Working with Rows

For working with rows, use `RowManager`. To get an item ID from a row ID, use `rowManager.getRow(rowId).getItemId()`. This gives you `ItemIdentity` instance. To see if it is an issue, use `CoreIdentities.isIssue(itemId)` and to get issue ID in that case, use `itemId.getLongId()`.

To get all row IDs for a given issue ID (for example, to find an issue in a forest), you can use `rowManager.findRows(CoreIdentities.issue(issueId))`. These row IDs may be from multiple forests, so you need to see if the forest that you have contains some of those IDs.

Getting Totals and Other Values

To calculate totals or other Structure-calculated values, you need to use `StructureAttributeService`.

`StructureAttributeService.getAttributeValues()` has the following parameters:

- `ForestSpec` – use the same forest spec that you use to retrieve the forest;
- row IDs – you need to specify for which rows (not issues!) the values are requested;
- a collection of `AttributeSpec` – specify which attributes are requested.

You need to build a list of attribute specs to specify what to calculate. There are several ways to get a correct attribute spec:

- Some specs are defined in `CoreAttributeSpecs`.
- You can build a spec using `AttributeSpecBuilder`.
- You can parse a JSON representation of a spec into a `Map`, then extract "id" and "params".

Examples:

Attribute	Spec
Story Points	`AttributeSpecBuilder .create(“customfield", ValueFormat.NUMBER) .params() .set("fieldId", 10000) // 10000 - the id of "Story Points" custom field .build()`
∑ Story Points	`AttributeSpecBuilder .create("sum", ValueFormat.NUMBER) .params() .setAttribute(storyPoints) // storyPoints = the attribute spec for Story Points .set("distinct", true) // exclude duplicates .build()`

Changing Structure

To change a structure, you need to use `UpdatableForestSource.apply()` method. Each update is a separate transaction – the concept of a ForestTransaction used in Structure 2 has been removed.

To get an instance of `UpdatableForestSource` you need to cast `ForestSource` retrieved from `ForestService`.

Examples:

Operation	Code
Add an issue with ID 10200 to structure, under parent row with ID 1040, after row with ID 1900 and before row with ID 2000	`forestSource.apply(new UpdatableForestSource.Update.Add( CoreIdentities.issue(10200), 1040, 1900, 2000))`
Remove rows with IDs 10100 and 10102	`forestSource.apply(new UpdatableForestSource.Update.Remove( LongArray.create(10100, 10102)))`
Move row with ID 1010 as the first row under parent row with ID 1040, before a row with ID 1060	`forestSource.apply(new UpdatableForestSource.Update.Move( LongArray.create(1010), 1040, 0, 1060))`

8.8.2 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[740] if you need help.

Structure comes with a number of bundled synchronizers(see page 514), but you can add another synchronizer to the system, allowing Structure users to install it on structures and run export / import.

740 mailto:support@almworks.com

8.8.2.1 Implement `StructureSynchronizer`

Create your implementation of StructureSynchronizer[741] interface. Use AbstractSynchronizer[742] as the base class.

8.8.2.2 Define `structure-synchronizer` Module

Add structure-synchronizer(see page 974) module to your `atlassian-plugin.xml`, referring to your implementation of the `StructureSynchronizer`.

8.8.2.3 Test Thoroughly

Test how your synchronizer works when other synchronizers are also installed onto the same structure.

8.8.2.4 Sample Project

This project can be used to bootstrap writing your own synchronizer. It compiles into a working plugin, which does not do anything except writing to console at the times the synchronizer would do some work.

You can download the sources zip with the sample synchronizers at API Usage Samples(see page 965) page.

8.8.3 structure-synchronizer

Synchronizer module allows you to plug additional synchronizers into Structure.

8.8.3.1 Module description sample

Here's a template of a synchronizer module declaration, and explanation of the parameters follows.

```
  <structure-synchronizer key="module-key" order="100"
                          class="com.company.your.plugin.sync.SyncClass">
    <label key="label.i18n.key">Name of Synchronizer</label>
    <description key="description.i18n.key">Description of Synchronizer</description>
    <rules key="rules.i18n.key">Large text to be shown at the top of synchronizer's
configuration page.</rules>
    <resource type="velocity" name="form" location="/templates/myplugin/sync-form.vm"
/>
  </structure-synchronizer>
```

741 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/StructureSynchronizer.html
742 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/AbstractSynchronizer.html

Element	Required	Description
`structure-synchronizer`	Yes	The module descriptor.
`@key`	Yes	Unique module key within the plugin.
`@order`	Yes	Order of the synchronizer among other synchronizers, whenever a list of synchronizers is present.
`@class`	Yes	The class that implements the synchronizer. Must implement StructureSynchronizer[743]. It is recommended to extend AbstractSynchronizer[744].
`label`	Yes	The name of the synchronizer.
`description`	No	Description of the synchronizer.
`rules`	No	The text that is shown at the top of the synchronizer configuration page. Could be a large text.
`resource[@name="form"]`	Yes	A velocity template that contains the form for the synchronizer parameters.

8.9 Open Source Licenses

Structure for Jira is made possible by open source software.

The following is a list of open source libraries used in the product and links to their respective license agreements.

743 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/StructureSynchronizer.html
744 http://almworks.com/structure/javadoc/latest/com/almworks/jira/structure/api/sync/AbstractSynchronizer.html

Component / Library	License
Annotations (JetBrains)[745]	Apache 2.0[746]
Apache Commons[747]	Apache 2.0[748]
Apache Derby[749]	Apache 2.0[750]
Apache HttpClient[751]	Apache 2.0[752]
Apache POI[753]	Apache 2.0[754]
HPPC[755]	Apache 2.0[756]
Jackson[757]	Apache 2.0[758]
JavaEWAH[759]	Apache 2.0[760]
Javolution[761]	BSD license[762]
Jsoup[763]	MIT license[764]
jQuery[765]	MIT license[766]
jQuery UI[767]	MIT license[768]
hoverIntent[769]	MIT license[770]

745 https://github.com/JetBrains/java-annotations
746 https://www.apache.org/licenses/LICENSE-2.0
747 https://commons.apache.org/
748 https://www.apache.org/licenses/LICENSE-2.0
749 https://db.apache.org/derby/
750 https://www.apache.org/licenses/LICENSE-2.0
751 https://hc.apache.org/httpcomponents-client-ga/
752 https://www.apache.org/licenses/LICENSE-2.0
753 https://poi.apache.org/
754 https://www.apache.org/licenses/LICENSE-2.0
755 https://labs.carrotsearch.com/hppc.html
756 https://www.apache.org/licenses/LICENSE-2.0
757 https://github.com/FasterXML/jackson
758 https://www.apache.org/licenses/LICENSE-2.0
759 https://github.com/lemire/javaewah
760 https://www.apache.org/licenses/LICENSE-2.0
761 http://javolution.org/
762 http://javolution.org/LICENSE.txt
763 https://jsoup.org/
764 https://jsoup.org/license
765 https://jquery.com/
766 https://jquery.org/license/
767 https://jqueryui.com/
768 https://jquery.org/license/
769 https://github.com/briancherne/jquery-hoverIntent
770 https://opensource.org/licenses/MIT

Component / Library	License
Moment.js[771]	MIT license[772]
Font Awesome[773]	SIL OFL 1.1, MIT license[774]

ALM Works is a proud participant of Pledge 1%.[775] As a part of our pledge to donate 1% of equity, product and time to the community, we're offering all our products free of charge to open-source projects and community projects. You can request a license through Atlassian's Open Source License Request[776] page or Community License Request[777] page. Should you have any questions or need assistance, please let us know at support@almworks.com[778].

771 https://momentjs.com/
772 https://github.com/moment/moment/blob/develop/LICENSE
773 https://fontawesome.com/
774 https://fontawesome.com/v4.7.0/license/
775 https://pledge1percent.org/
776 https://www.atlassian.com/software/views/open-source-license-request
777 https://www.atlassian.com/software/views/community-license-request
778 mailto:support@almworks.com

9 Download

9.1 Download Structure Plugin

Please select the version appropriate for your Jira.

Structure version	Jira version	Download link	Build	Publish Date	End of Life	md5sum
6.4	Jira 7.13+	structure-6.4.0.jar[779]	36909	2020-12-07	2021-12-07	187461a726b301bdefd3e86ec5d12375

9.2 Download Structure Extensions

Extension	Version	Compatibility	Download link	Publish Date	End of Life
Structure.Testy	2.4	Jira 7.13+ Structure 6.0+	testy-2.4.jar[780] md5 92e1714d9a5a2187d6482e83293040e8	2020-03-26	2021-03-26
Structure.Pages Confluence Helper for Structure.Pages	See Structure.Pages Download[781]				

End-User License Agreement (PDF)[782]

End of Life:

- For beta and release candidate versions: Beta versions may be scheduled to expire after a certain amount of time. In any case, the lifespan of such a pre-release version is not supposed to be longer than 3 months. You will need to upgrade to a newer version before the end-of-life date.
- For other versions: You will be able to use this version of the plugin indefinitely; however, after the end-of-life date, the support for the version is limited.

779 https://d1.almworks.com/.files/structure-6.4.0.jar
780 https://d1.almworks.com/.files/testy-2.4.0.jar
781 https://wiki.almworks.com/display/pages/Download
782 http://almworks.com/EULA-Structure.pdf

9.3 What's Next?

See Structure 6.3 Release Notes(see page 577) and Getting Started with Structure(see page 49)

9.4 Documentation

Links to the available documentation collections:

Download documentation:

File	Modified
Structure-Documentation-v6_3_1.xml.zip[783]	Oct 10, 2020 by Konstantin Romanovsky[784]
Structure-Documentation-v6_3_1.pdf[785]	Oct 10, 2020 by Konstantin Romanovsky[786]

9.5 Download Archive

This page contains links to older version of the Structure plugin.

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
6.4	Jira 7.13+	structure-6.4.0.jar[787]	36909	2020-12-07	2021-12-07	187461a726b301bdefd3e86ec5d12375
6.3.2	Jira 7.13+	structure-6.3.2.jar[788]	36567	2020-11-18	2021-11-18	aa16b82b6a981185939d5748ff7d3e73
6.3.1	Jira 7.13+	structure-6.3.1.jar[789]	35873	2020-10-10	2021-10-10	1c07db50c8b0a4f4f42d1a5066816e50

783 https://wiki.almworks.com/download/attachments/32222954/Structure-Documentation-v6_3_1.xml.zip?api=v2
784 https://wiki.almworks.com/display/~kromanovsky
785 https://wiki.almworks.com/download/attachments/32222954/Structure-Documentation-v6_3_1.pdf?api=v2
786 https://wiki.almworks.com/display/~kromanovsky
787 https://d1.almworks.com/.files/structure-6.4.0.jar
788 https://d1.almworks.com/.files/structure-6.3.2.jar
789 https://d1.almworks.com/.files/structure-6.3.1.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
6.3	Jira 7.13+	structure-6.3.0.jar[790]	35796	2020-10-08	2021-10-08	4f3f450bc18d1196f37696e27223975e
6.2.1	Jira 7.13+	structure-6.2.1.jar[791]	35029	2020-09-02	2021-09-02	a4bdd94aeb6a0e06bfc9eab775b66dae
6.2	Jira 7.13+	structure-6.2.0.jar[792]	34542	2020-08-10	2021-08-10	70b5e695ed8f13bee1e7bed4a5f165c9
6.1.2	Jira 7.13+	structure-6.1.2.jar[793]	33945	2020-07-06	2021-07-06	5b45cc3ace69ff03352576c407892482
6.1.1	Jira 7.13+	structure-6.1.1.jar[794]	33659	2020-06-17	2021-06-17	8acdff591068c6b48711539fd2ffc7b3
6.1	Jira 7.13+	structure-6.1.0.jar[795]	33417	2020-06-03	2021-06-03	b58eac7d8e02a77492907bd95616aaf2
6.0.1	Jira 7.13+	structure-6.0.1.jar[796]	32793	2020-05-05	2021-05-05	0b2a035debd070f9a41135bf2b003bdb
6.0	Jira 7.13+	structure-6.0.0.jar[797]	32050	2020-03-26	2021-03-26	a1ba401d9eb311f41a1ac2c8ef643524

790 https://d1.almworks.com/.files/structure-6.3.0.jar
791 https://d1.almworks.com/.files/structure-6.2.1.jar
792 https://d1.almworks.com/.files/structure-6.2.0.jar
793 https://d1.almworks.com/.files/structure-6.1.2.jar
794 https://d1.almworks.com/.files/structure-6.1.1.jar
795 https://d1.almworks.com/.files/structure-6.1.0.jar
796 https://d1.almworks.com/.files/structure-6.0.1.jar
797 https://d1.almworks.com/.files/structure-6.0.0.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
5.6.4	Jira 7.6+	structure-5.6.4.jar[798]	31000	2020-05-05	2021-05-05	33acfaa18df27b3ed98e28e666693e48
5.6.3	Jira 7.6+	structure-5.6.3.jar[799]	30600	2020-01-24	2021-01-24	f182eb51327d57eefdfcdfa9bdee0a1b
5.6.2	Jira 7.6+	structure-5.6.2.jar[800]	30177	2019-12-24	2020-12-24	a77fb1c7ae3edbd343b9ae046393af7e
5.6.1	Jira 7.6+	structure-5.6.1.jar[801]	28633	2019-10-14	2020-10-14	d58768e59ed9990c5a9f5cb3fa2a7d0a
5.6	Jira 7.6+	structure-5.6.0.jar[802]	28297	2019-09-19	2020-09-19	3c70f95463289ce478e576697205be86
5.5.1	Jira 7.6+	structure-5.5.1.jar[803]	27391	2019-08-15	2020-08-15	69cde87aafcec1b1009db14a04efee1a
5.5	Jira 7.6+	structure-5.5.0.jar[804]	26890	2019-07-18	2020-07-18	cfdae3d1a6940d0bdd42689ea51e0867

798 https://d1.almworks.com/.files/structure-5.6.4.jar
799 https://d1.almworks.com/.files/structure-5.6.3.jar
800 https://d1.almworks.com/.files/structure-5.6.2.jar
801 https://d1.almworks.com/.files/structure-5.6.1.jar
802 https://d1.almworks.com/.files/structure-5.6.0.jar
803 https://d1.almworks.com/.files/structure-5.5.1.jar
804 https://d1.almworks.com/.files/structure-5.5.0.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
5.4.1	Jira 7.6+	structure-5.4.1.jar[805]	26728	2019-07-05	2020-07-05	54ea92f79d656e507a08066d6166966d
5.4	Jira 7.6+	structure-5.4.0.jar[806]	26352	2019-06-05	2020-06-05	e5b78ce8995c9b4f4cc666fba316ebb6
5.3	Jira 7.6+	structure-5.3.0.jar[807]	25522	2019-03-26	2020-03-26	092763ea41d479e4ed9071ba6892eb6d
5.2.2	Jira 7.6+	structure-5.2.2.jar[808]	25109	2019-03-01	2020-03-01	4e80a6e6546aeaba4c9111edc0bfd83d
5.2.1	Jira 7.6+	structure-5.2.1.jar[809]	24822	2019-02-06	2020-02-06	a435400d6acc22013921f5160d62aa58
5.2	Jira 7.6+	structure-5.2.0.jar[810]	24591	2018-12-27	2019-12-27	9c22f0934b91ae85484aa18ed70870d1
5.1.1	Jira 7.2+	structure-5.1.1.jar[811]	24000	2020-05-05	2021-05-05	bdca2f7b51233164bc9fb5219462a6b3
5.1	Jira 7.2+	structure-5.1.0.jar[812]	23905	2018-10-25	2019-10-25	1e1d7dfd3246b3b88415f2463222ffdc

805 https://d1.almworks.com/.files/structure-5.4.1.jar
806 https://d1.almworks.com/.files/structure-5.4.0.jar
807 https://d1.almworks.com/.files/structure-5.3.0.jar
808 https://d1.almworks.com/.files/structure-5.2.2.jar
809 https://d1.almworks.com/.files/structure-5.2.1.jar
810 https://d1.almworks.com/.files/structure-5.2.0.jar
811 https://d1.almworks.com/.files/structure-5.1.1.jar
812 https://d1.almworks.com/.files/structure-5.1.0.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
5.0	Jira 7.2+	structure-5.0.0.jar[813]	23071	2018-08-16	2019-08-16	05a4109afbdea54a7e877ef5140de500
4.6.5	Jira 7.2+	structure-4.6.5.jar[814]	21932	2018-06-13	2019-06-13	a2065e24f6cbd67960698bafd15581db
4.6.3	Jira 7.2+	structure-4.6.3.jar[815]	21429	2018-05-14	2019-05-14	fc99383bd524318f6d14b6f86d408b0b
4.6.1	Jira 7.2+	structure-4.6.1.jar[816]	21239	2018-04-27	2019-04-27	f0fe1fd44015b2cba9024cfba1e8e438
4.6.0	Jira 7.2+	structure-4.6.0.jar[817]	20817	2018-03-27	2019-03-27	e582186b122c48c66ce123da25d37ce9
4.5.1	Jira 7.2+	structure-4.5.1.jar[818]	20256	2018-01-19	2019-01-19	fdb7abfe158dcae24005621ab014fd67
4.5.0	Jira 7.2+	structure-4.5.0.jar[819]	19994	2017-12-28	2018-12-28	543ed2f073f85710af33e109f9ac1dea
4.4.0	Jira 7.2+	structure-4.4.0.jar[820]	19514	2017-11-29	2018-11-29	df72ef8c8a52257d1b9a3d6085cc0873
4.3.0	Jira 7.2+	structure-4.3.0.jar[821]	19053	2017-10-20	2018-10-20	47bc93338711b19250ed98e53eeced25
4.2.0	Jira 7.2+	structure-4.2.0.jar[822]	18430	2017-08-25	2018-08-25	56bf4cb46edb400d9e5e0ec71a89161b

813 https://d1.almworks.com/.files/structure-5.0.0.jar
814 https://d1.almworks.com/.files/structure-4.6.5.jar
815 https://d1.almworks.com/.files/structure-4.6.3.jar
816 https://d1.almworks.com/.files/structure-4.6.1.jar
817 https://d1.almworks.com/.files/structure-4.6.0.jar
818 https://d1.almworks.com/.files/structure-4.5.1.jar
819 https://d1.almworks.com/.files/structure-4.5.0.jar
820 https://d1.almworks.com/.files/structure-4.4.0.jar
821 https://d1.almworks.com/.files/structure-4.3.0.jar
822 https://d1.almworks.com/.files/structure-4.2.0.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
4.1.2	Jira 7.1+	structure-4.1.2.jar[823]	19000	2020-05-05	2021-05-05	1f338de829e7dfdd3152aca3c15f1c37
4.1.1	Jira 7.1+	structure-4.1.1.jar[824]	18995	2017-10-20	2018-10-20	8aa826eeae801fd022252f8274172184
4.1.0	Jira 7.1+	structure-4.1.0.jar[825]	17626	2017-06-19	2018-06-19	975594ce5358b2f1d8d5843ce060c790
4.0.0	Jira 7.1+	structure-4.0.0.jar[826]	17063	2017-04-26	2018-04-26	4cfed9da9885477d3a16576ab9559057
3.6.0	Jira 7.1+	structure-3.6.0.jar[827]	16687	2017-04-04	2018-04-04	0fe8a3e02429840e7331ddebb2f7c802
3.5.2	Jira 7.0+	structure-3.5.2.jar[828]	17500	2020-05-05	2021-05-05	43fe189880fa7b97ad551a2b94a2932f
3.5.1	Jira 7.0+	structure-3.5.1.jar[829]	17651	2017-06-19	2018-06-19	f7bd5d660a0b66df0decf6b6ac2e8e6c
3.5.0	Jira 7.0+	structure-3.5.0.jar[830]	16134	2017-01-30	2018-01-30	8ef314fae4039e9fa3aaca82b872c413
3.4.1	Jira 7.0+	structure-3.4.1.jar[831]	15834	2016-12-30	2017-12-30	54a0bb98afb025fe1ec9c5453316cd55
3.4.0	Jira 7.0+	structure-3.4.0.jar[832]	15635	2016-12-12	2017-12-12	e5ab3ded92f25e58ea442723a2643c40

823 https://d1.almworks.com/.files/structure-4.1.2.jar
824 https://d1.almworks.com/.files/structure-4.1.1.jar
825 https://d1.almworks.com/.files/structure-4.1.0.jar
826 https://d1.almworks.com/.files/structure-4.0.0.jar
827 https://d1.almworks.com/.files/structure-3.6.0.jar
828 https://d1.almworks.com/.files/structure-3.5.2.jar
829 https://d1.almworks.com/.files/structure-3.5.1.jar
830 https://d1.almworks.com/.files/structure-3.5.0.jar
831 https://d1.almworks.com/.files/structure-3.4.1.jar
832 https://d1.almworks.com/.files/structure-3.4.0.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
3.3.5.jira6	Jira 6.4.x	structure-3.3.5.jira6.jar[833]	17646	2017-06-19	2018-06-19	d1feb277d8c071296df9e4176195632d
3.3.4.jira6	Jira 6.4.x	structure-3.3.4.jira6.jar[834]	16129	2017-01-30	2018-01-30	fb12064eb4c81af53cdbe674176cc7c2
3.3.3.jira7	Jira 7.0+	structure-3.3.3.jira7.jar[835]	15283	2016-11-15	2017-11-15	ebab71cf97a9768db69b39b39e958382
3.3.3.jira6	Jira 6.4.x	structure-3.3.3.jira6.jar[836]	15280	2016-11-15	2017-11-15	303e275eac1196732a4bb44680f10fe7
3.3.2.jira7	Jira 7.0+	structure-3.3.2.jira7.jar[837]	14573	2016-10-03	2017-10-03	1dc7fb3bf533c6e47bec6c890376a29e
3.3.2.jira6	Jira 6.4.x	structure-3.3.2.jira6.jar[838]	14570	2016-10-03	2017-10-03	eea5925cbf738e8fed62be657a9e1f2d
3.3.1.jira7	Jira 7.0+	structure-3.3.1.jira7.jar[839]	14273	2016-09-22	2017-09-22	e11d7d726b1f78b1d632407d39254014
3.3.1.jira6	Jira 6.4.x	structure-3.3.1.jira6.jar[840]	14271	2016-09-22	2017-09-22	312e192ab68cc1086388b7bf5f5cc8b1
3.3.0.jira7	Jira 7.0+	structure-3.3.0.jira7.jar[841]	13804	2016-09-09	2017-09-09	a62f6588ddc5f090724f1dcb5a87f6ae
3.3.0.jira6	Jira 6.4.x	structure-3.3.0.jira6.jar[842]	13802	2016-09-09	2017-09-09	8cb038673694bcbe27ca07fd736ac8e4
3.2.3.jira6	Jira 6.3-6.4.x	structure-3.2.3.jira6.jar[843]	17642	2017-06-19	2018-06-19	703ef49c587f688d9989e8b3032a5c87

833 https://d1.almworks.com/.files/structure-3.3.5.jira6.jar
834 https://d1.almworks.com/.files/structure-3.3.4.jira6.jar
835 https://d1.almworks.com/.files/structure-3.3.3.jira7.jar
836 https://d1.almworks.com/.files/structure-3.3.3.jira6.jar
837 https://d1.almworks.com/.files/structure-3.3.2.jira7.jar
838 https://d1.almworks.com/.files/structure-3.3.2.jira6.jar
839 https://d1.almworks.com/.files/structure-3.3.1.jira7.jar
840 https://d1.almworks.com/.files/structure-3.3.1.jira6.jar
841 https://d1.almworks.com/.files/structure-3.3.0.jira7.jar
842 https://d1.almworks.com/.files/structure-3.3.0.jira6.jar
843 https://d1.almworks.com/.files/structure-3.2.3.jira6.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
3.2.2.jira6	Jira 6.3-6.4.x	structure-3.2.2.jira6.jar[844]	16127	2017-01-30	2018-01-30	f1dab1e8a99ed88114a759e2284e5441
3.2.1.jira7	Jira 7.0+	structure-3.2.1.jira7.jar[845]	12715	2016-07-05	2017-07-05	25d9e2171a9de36cec603b6a17ec7525
3.2.1.jira6	Jira 6.3 – 6.4.x	structure-3.2.1.jira6.jar[846]	12713	2016-07-05	2017-07-05	3d4d50d920280c78d6e982011fba8b66
3.2.0.jira7	Jira 7.0+	structure-3.2.0.jira7.jar[847]	12444	2016-06-18	2017-06-18	c4e63140c4fb86798c05fc7f2c032b36
3.2.0.jira6	Jira 6.3 – 6.4.x	structure-3.2.0.jira6.jar[848]	12442	2016-06-18	2017-06-18	2d9d4f42a7e55e312e9e12ebf0f3af5f
3.1.1.jira7	Jira 7.0+	structure-3.1.1.jira7.jar[849]	12147	2016-06-02	2017-06-02	cd355ec9ff95e335f9e7d587e48335dd
3.1.1.jira6	Jira 6.3 – 6.4.x	structure-3.1.1.jira6.jar[850]	12144	2016-06-02	2017-06-02	8aaaefbbec05f1d4a81d6ddc382a6ffa
3.1.0.jira7	Jira 7.0+	structure-3.1.0.jira7.jar[851]	11645	2016-04-30	2017-04-30	5110b83c9ea8601a0e2a64eee292769a
3.1.0.jira6	Jira 6.3 – 6.4.x	structure-3.1.0.jira6.jar[852]	11644	2016-04-30	2017-04-30	f87b93ea163ae33f6a920a3171ed148a
3.0.2.jira7	Jira 7.0+	structure-3.0.2.jira7.jar[853]	11528	2016-04-26	2017-04-26	90b8f5f8f900a3498baa09493cdbc1bd
3.0.2.jira6	Jira 6.3 – 6.4.x	structure-3.0.2.jira6.jar[854]	11526	2016-04-26	2017-04-26	1bbb2615d76091022f00e07be5d03693

844 https://d1.almworks.com/.files/structure-3.2.2.jira6.jar
845 https://d1.almworks.com/.files/structure-3.2.1.jira7.jar
846 https://d1.almworks.com/.files/structure-3.2.1.jira6.jar
847 https://d1.almworks.com/.files/structure-3.2.0.jira7.jar
848 https://d1.almworks.com/.files/structure-3.2.0.jira6.jar
849 https://d1.almworks.com/.files/structure-3.1.1.jira7.jar
850 https://d1.almworks.com/.files/structure-3.1.1.jira6.jar
851 https://d1.almworks.com/.files/structure-3.1.0.jira7.jar
852 https://d1.almworks.com/.files/structure-3.1.0.jira6.jar
853 https://d1.almworks.com/.files/structure-3.0.2.jira7.jar
854 https://d1.almworks.com/.files/structure-3.0.2.jira6.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
3.0.1.jira7	Jira 7.0+	structure-3.0.1.jira7.jar[855]	11315	2016-04-11	2017-04-11	929fcfdac6c4e939596499f2d11535cc
3.0.1.jira6	Jira 6.3 – 6.4.x	structure-3.0.1.jira6.jar[856]	11314	2016-04-11	2017-04-11	9ffc3f78d653f22564b6badd6c226ffb
3.0.0.jira7	Jira 7.0+	structure-3.0.0.jira7.jar[857]	11142	2016-04-05	2017-04-05	3937e3c80511bcbf4cd11d551335d2fc
3.0.0.jira6	Jira 6.3 – 6.4.x	structure-3.0.0.jira6.jar[858]	11141	2016-04-05	2017-04-05	ce7590fa0af69b9d031345fbb0869a15
2.11.2.jira7	Jira 7.0+	structure-2.11.2.jira7.jar[859]	10115	2016-02-25	2017-02-25	9b182941865f816229d7bcb7a4a0f251
2.11.2.jira6	Jira 6.3 – 6.4.x	structure-2.11.2.jira6.jar[860]	10113	2016-02-25	2017-02-25	44d8c2c693b4f8e10aeec2bc8fd9392f
2.11.1.jira7	JIRA 7.0+	structure-2.11.1.jira7.jar[861]	8484	2015-12-16	2016-12-16	684e320a7d7ad6b2974cc403e34fd84b
2.11.1.jira6	JIRA 6.3 – 6.4.x	structure-2.11.1.jira6.jar[862]	8483	2015-12-16	2016-12-16	7846f40430c1170b0d00c72d3758bb5d
2.11.0.jira7	JIRA 7.0+	structure-2.11.0.jira7.jar[863]	7634	2015-10-16	2016-10-16	bbe3492315892c5d603396ea2a0f4f58
2.11.0.jira6	JIRA 6.3 – 6.4.x	structure-2.11.0.jira6.jar[864]	7635	2015-10-16	2016-10-16	6a6ac210e7ed862be62cc1e71e5adbdd
2.10.6	JIRA 6.1 – 6.4.x	structure-2.10.6.jar[865]	6197	2015-07-03	2016-07-03	61947f5b97217d31d2926104231644e1
2.10.5	JIRA 6.1 – 6.4.x	structure-2.10.5.jar[866]	5964	2015-06-04	2016-06-04	44849ddcc940dd34b6cf862df91d43cd

855 https://d1.almworks.com/.files/structure-3.0.1.jira7.jar
856 https://d1.almworks.com/.files/structure-3.0.1.jira6.jar
857 https://d1.almworks.com/.files/structure-3.0.0.jira7.jar
858 https://d1.almworks.com/.files/structure-3.0.0.jira6.jar
859 https://d1.almworks.com/.files/structure-2.11.2.jira7.jar
860 https://d1.almworks.com/.files/structure-2.11.2.jira6.jar
861 https://d1.almworks.com/.files/structure-2.11.1.jira7.jar
862 https://d1.almworks.com/.files/structure-2.11.1.jira6.jar
863 https://d1.almworks.com/.files/structure-2.11.0.jira7.jar
864 https://d1.almworks.com/.files/structure-2.11.0.jira6.jar
865 https://d1.almworks.com/.files/structure-2.10.6.jar
866 https://d1.almworks.com/.files/structure-2.10.5.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
2.10.4	JIRA 6.1 – 6.4.x	structure-2.10.4.jar[867]	5880	2015-05-29	2016-05-29	bb435e06f769df66981000751b9b09e4
2.10.3	JIRA 6.1 – 6.4.x	structure-2.10.3.jar[868]	5688	2015-04-30	2016-04-30	1b4e3ce3b2c335a4fe05b557f4af9e65
2.10.2	JIRA 6.1 – 6.4.x	structure-2.10.2.jar[869]	5663	2015-04-27	2016-04-27	f03c9ee599c28d21a96a0d400d7dcf9c
2.10.1	JIRA 6.1 – 6.4.x	structure-2.10.1.jar[870]	5467	2015-03-26	2016-03-26	5f489fe1dcc491075f8c496e31e8c283
2.10.0	JIRA 6.1 – 6.3.15	structure-2.10.0.jar[871]	5206	2015-02-23	2016-02-23	c943296433b01fe8ab74d25fffc44ff5
2.9.1	JIRA 6.1 – 6.3+	structure-2.9.1.jar[872]	4572	2014-10-28	2015-10-28	1fb9cdf5d612c8ce9711293b98c299ea
2.9.0	JIRA 6.1 – 6.3+	structure-2.9.0.jar[873]	4460	2014-09-08	2015-09-08	b03dfb5081faf76c9fd4fe043e2d12bd
2.8.3	JIRA 6.1 – 6.3+	structure-2.8.3.jar[874]	4228	2014-07-25	2015-07-25	c54909cc9bc02e4c41caf76f622ff90e
2.8.2	JIRA 6.1 – 6.3+	structure-2.8.2.jar[875]	4214	2014-07-21	2015-07-21	e14ecebefb954b8275c3096fab7568af
2.8.1	JIRA 6.1 – 6.2.7	structure-2.8.1.jar[876]	4191	2014-07-17	2015-07-17	40a021f938154a5bd38e4f66151c9a40

867 https://d1.almworks.com/.files/structure-2.10.4.jar
868 https://d1.almworks.com/.files/structure-2.10.3.jar
869 https://d1.almworks.com/.files/structure-2.10.2.jar
870 https://d1.almworks.com/.files/structure-2.10.1.jar
871 https://d1.almworks.com/.files/structure-2.10.0.jar
872 https://d1.almworks.com/.files/structure-2.9.1.jar
873 https://d1.almworks.com/.files/structure-2.9.0.jar
874 https://d1.almworks.com/.files/structure-2.8.3.jar
875 https://d1.almworks.com/.files/structure-2.8.2.jar
876 https://d1.almworks.com/.files/structure-2.8.1.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
2.8.0	JIRA 6.1 – 6.3+	structure-2.8.0.jar[877]	4151	2014-07-07	2015-07-07	ba56673401c7a818297b153a23c2ca19
2.7.0	JIRA 6.0 – 6.2+	structure-2.7.0.jar[878]	3850	2014-02-26	2015-02-26	7d59559069ee957848452e9dee569ab7
2.6.1	JIRA 5.2 – 5.2.11	structure-2.6.1.jira5.jar[879]	3810	2014-02-25	2015-02-25	595b6a723eb3e1cd3a70f38ff7bef81b
2.6.0	JIRA 6.0 – 6.1+	structure-2.6.0.jar[880]	3701	2014-01-24	2015-01-24	dc5a9d344d7ace5176f7f1cde6bd6e9c
2.6.0	JIRA 5.2 – 5.2.11	structure-2.6.0.jira5.jar[881]	3698	2014-01-24	2015-01-24	ee4a0b0c8e91c122503b3dc7554b9815
2.5.3	JIRA 6.0 – 6.1+	structure-2.5.3.jar[882]	3496	2013-11-22	2014-11-22	df35dc5440d6287402bd0afabf7afb9f
2.5.3	JIRA 5.2 – 5.2.11	structure-2.5.3.jira5.jar[883]	3493	2013-11-22	2014-11-22	24232bf5572ceeb0b9b9af7a56214340
2.5.2	JIRA 6.0 – 6.1+	structure-2.5.2.jar[884]	3455	2013-11-01	2014-11-01	76e72d8384df35c2e3f4e077f4621d8e
2.5.2	JIRA 5.2 – 5.2.11	structure-2.5.2.jira5.jar[885]	3452	2013-11-01	2014-11-01	6481fee6eeeb3a817fb510a582983540

877 https://d1.almworks.com/.files/structure-2.8.0.jar
878 https://d1.almworks.com/.files/structure-2.7.0.jar
879 https://d1.almworks.com/.files/structure-2.6.1.jira5.jar
880 https://d1.almworks.com/.files/structure-2.6.0.jar
881 https://d1.almworks.com/.files/structure-2.6.0.jira5.jar
882 https://d1.almworks.com/.files/structure-2.5.3.jar
883 https://d1.almworks.com/.files/structure-2.5.3.jira5.jar
884 https://d1.almworks.com/.files/structure-2.5.2.jar
885 https://d1.almworks.com/.files/structure-2.5.2.jira5.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
2.5.1	JIRA 6.0 – 6.1+	structure-2.5.1.jar[886]	3429	2013-10-30	2014-10-30	6b460040bf22eced9f03f0f4fe7133d7
2.5.1	JIRA 5.2 – 5.2.11	structure-2.5.1.jira5.jar[887]	3427	2013-10-30	2014-10-30	7322b4d016451a8a2883efb79d545219
2.5.0	JIRA 6.0 – 6.1+	structure-2.5.0.jar[888]	3273	2013-09-26	2014-09-26	a0c6110c7696f8f95241480e5acfe509
2.5.0	JIRA 5.2 – 5.2.11	structure-2.5.0.jira5.jar[889]	3271	2013-09-26	2014-09-26	e6d010d9f8d77127c58ac086be30746a
2.4.3	JIRA 6.0+	structure-2.4.3.jar[890]	3096	2013-09-01	2014-09-01	aa8351adff97070088503e8fbb89b9dd
2.4.3	JIRA 5.0.1 – 5.2+	structure-2.4.3.jira5.jar[891]	3093	2013-09-01	2014-09-01	f6a9c660777e92a67cc3ad704e4f9304
2.4.2	JIRA 6.0+	structure-2.4.2.jar[892]	3038	2013-08-07	2014-08-07	548b80001bcdb3c9df1ba9c0ddaa687d
2.4.2	JIRA 5.0.1 – 5.2+	structure-2.4.2.jira5.jar[893]	3036	2013-08-07	2014-08-07	aaa4a979b3da658806b87b5d260e4517
2.4.1	JIRA 6.0+	structure-2.4.1.jar[894]	3018	2013-08-02	2014-08-02	8f7e92dad13a40a8bc630b26dae956da

886 https://d1.almworks.com/.files/structure-2.5.1.jar
887 https://d1.almworks.com/.files/structure-2.5.1.jira5.jar
888 https://d1.almworks.com/.files/structure-2.5.0.jar
889 https://d1.almworks.com/.files/structure-2.5.0.jira5.jar
890 https://d1.almworks.com/.files/structure-2.4.3.jar
891 https://d1.almworks.com/.files/structure-2.4.3.jira5.jar
892 https://d1.almworks.com/.files/structure-2.4.2.jar
893 https://d1.almworks.com/.files/structure-2.4.2.jira5.jar
894 https://d1.almworks.com/.files/structure-2.4.1.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
2.4.1	JIRA 5.0.1 – 5.2+	structure-2.4.1.jira5.jar[895]	3017	2013-08-02	2014-08-02	ee875d01f47c57bea1f2770607b3edec
2.4.0	JIRA 6.0+	structure-2.4.0.jar[896]	2782	2013-06-14	2014-06-14	66fa84503c5e787d4643a8809754daac
2.4.0	JIRA 5.0.1 – 5.2+	structure-2.4.0.jira5.jar[897]	2772	2013-06-14	2014-06-14	ad5f88295185af4642083efc10a6b27d
2.3.0	JIRA 6.0+	structure-2.3.0.jar[898]	2686	2013-05-18	2014-05-18	329921b60229a0f623b36001e7c4f67c
2.3.0	JIRA 5.0 – 5.2+	structure-2.3.0.jira5.jar[899]	2681	2013-05-18	2014-05-18	deae862ca01f0ff02103387a4796d371
2.2.1	JIRA 5.0 – 5.2+	structure-2.2.1.jar[900]	2524	2013-04-12	2014-04-12	83b6f08e131c95c5298b188bd08e01a4
2.2.0	JIRA 5.0 – 5.2+	structure-2.2.0.jar[901]	2458	2013-03-28	2014-03-28	66e9d6340f31a7c245866ef8849ea8ea
2.1.1	JIRA 5.0 – 5.2+	structure-2.1.1.jar[902]	2405	2013-03-18	2014-03-18	309396fef173778035bfe60d1d210c61
2.1.0	JIRA 5.0 – 5.2+	structure-2.1.0.jar[903]	2312	2013-02-15	2014-02-15	fff88a32d389171280a39c5792636089

895 https://d1.almworks.com/.files/structure-2.4.1.jira5.jar
896 https://d1.almworks.com/.files/structure-2.4.0.jar
897 https://d1.almworks.com/.files/structure-2.4.0.jira5.jar
898 https://d1.almworks.com/.files/structure-2.3.0.jar
899 https://d1.almworks.com/.files/structure-2.3.0.jira5.jar
900 https://d1.almworks.com/.files/structure-2.2.1.jar
901 https://d1.almworks.com/.files/structure-2.2.0.jar
902 https://d1.almworks.com/.files/structure-2.1.1.jar
903 https://d1.almworks.com/.files/structure-2.1.0.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
2.0.0	JIRA 5.0 – 5.2+	structure-2.0.0.jar[904]	2054	2012-11-19	2013-11-19	fb08820d0bad6797219343f9ce6e40e4
2.0.0	JIRA 4.4	structure-2.0.0.jira44.jar[905]	2049	2012-11-19	2013-11-19	54343d964ccbf2107ab43d0e27a1da20
1.7.1	JIRA 5.0 – 5.1	structure-1.7.1.jar[906]	1646	2012-08-07	2013-08-07	cc4dd0808905ed27750e9b1d23ba8524
1.7.0	JIRA 5.0 – 5.1	structure-1.7.0.jar[907]	1544	2012-07-11	2013-07-11	2ef3a16f31ad6974c53a5104002dab0b
1.7.0	JIRA 4.4	structure-1.7.0.jira44.jar[908]	1532	2012-07-11	2013-07-11	1abbaf16cd918a9a77df54210151be14
1.6.0	JIRA 5.0	structure-1.6.0.jar[909]	1439	2012-05-12	2013-05-12	72faf0db1e10ddf9da4c34be0e8abc57
1.6.0	JIRA 4.4	structure-1.6.0.jira44.jar[910]	1433	2012-05-12	2013-05-12	0d7ce053e6114c73f36fee4419b7e8c0
1.6.0	JIRA 4.3	structure-1.6.0.jira43.jar[911]	1418	2012-05-12	2013-05-12	73cd71589c0852cf175a1f1ca5f7ba54
1.5.2	JIRA 5.0	structure-1.5.2.jar[912]	1301	2012-03-16	2013-03-16	9a456039c086212b16a946411fea1f2e

904 https://d1.almworks.com/.files/structure-2.0.0.jar
905 https://d1.almworks.com/.files/structure-2.0.0.jira44.jar
906 https://d1.almworks.com/.files/structure-1.7.1.jar
907 https://d1.almworks.com/.files/structure-1.7.0.jar
908 https://d1.almworks.com/.files/structure-1.7.0.jira44.jar
909 https://d1.almworks.com/.files/structure-1.6.0.jar
910 https://d1.almworks.com/.files/structure-1.6.0.jira44.jar
911 https://d1.almworks.com/.files/structure-1.6.0.jira43.jar
912 https://d1.almworks.com/.files/structure-1.5.2.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
1.5.2	JIRA 4.4	structure-1.5.2.jira44.jar[913]	1298	2012-03-16	2013-03-16	23123dbcbdb4eff3f42668622d34ad59
1.5.2	JIRA 4.3	structure-1.5.2.jira43.jar[914]	1295	2012-03-16	2013-03-16	e1fbeee8bde69e580e53fda801126fb5
1.5.1	JIRA 5.0	structure-1.5.1.jar[915]	1262	2012-02-22	2013-02-22	d6552ca8da9366268b30bbf2b4505747
1.5.0	JIRA 5.0	structure-1.5.0.jar[916]	1253	2012-02-17	2013-02-17	e82766909cb1eeed255e051045c56166
1.5.0	JIRA 4.4	structure-1.5.0.jira44.jar[917]	1248	2012-02-17	2013-02-17	4f753e1700836d2f1eb61e99bd756bcb
1.5.0	JIRA 4.3	structure-1.5.0.jira43.jar[918]	1242	2012-02-17	2013-02-17	9335abcec6f435385a2f82a029148bc6
1.4.1	JIRA 4.4	structure-1.4.1.jar[919]	1161	2012-01-16	2013-01-16	838276cf9ae777a5f4e072df56fad7fb
1.4.1	JIRA 4.3	structure-1.4.1.jira43.jar[920]	1158	2012-01-16	2013-01-16	c15bd779e874704d624cff68eed54bf5
1.4.1	JIRA 4.2	structure-1.4.1.jira42.jar[921]	1155	2012-01-16	2013-01-16	c4b8cd057128dd66ba293b738947d57e

913 https://d1.almworks.com/.files/structure-1.5.2.jira44.jar
914 https://d1.almworks.com/.files/structure-1.5.2.jira43.jar
915 https://d1.almworks.com/.files/structure-1.5.1.jar
916 https://d1.almworks.com/.files/structure-1.5.0.jar
917 https://d1.almworks.com/.files/structure-1.5.0.jira44.jar
918 https://d1.almworks.com/.files/structure-1.5.0.jira43.jar
919 https://d1.almworks.com/.files/structure-1.4.1.jar
920 https://d1.almworks.com/.files/structure-1.4.1.jira43.jar
921 https://d1.almworks.com/.files/structure-1.4.1.jira42.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
1.4	JIRA 4.4	structure-1.4.0.jar[922]	1138	2012-01-08	2013-01-08	cbf9a5b32305b252ad591e7a0722cb1f
1.4	JIRA 4.3	structure-1.4.0.jira43.jar[923]	1135	2012-01-08	2013-01-08	01d0bf95e1f40a70365b2b260d92d360
1.4	JIRA 4.2	structure-1.4.0.jira42.jar[924]	1132	2012-01-08	2013-01-08	1bfbda79f93e24ea55257c29ec7a1f60
1.3.2	JIRA 4.4	structure-1.3.2.jar[925]	1006	2011-11-29	2012-11-29	091bbea96a1f757e86b47ab6302ee506
1.3.2	JIRA 4.3	structure-1.3.2.jira43.jar[926]	1002	2011-11-29	2012-11-29	38a7481dc204046e6c52956e79a4ab14
1.3.2	JIRA 4.2	structure-1.3.2.jira42.jar[927]	998	2011-11-29	2012-11-29	bafb5476b1f5c73486d48a21e961d175
1.3.1	JIRA 4.4	structure-1.3.1.jar[928]	971	2011-11-10	2012-11-10	d426687615b34014ddace26aded9588a
1.3.1	JIRA 4.3	structure-1.3.1.jira43.jar[929]	968	2011-11-10	2012-11-10	a778f7f72f6a76d364064121fc3c02fb
1.3.1	JIRA 4.2	structure-1.3.1.jira42.jar[930]	963	2011-11-10	2012-11-10	51048202d1671cf4b4a0ffeadd9d89a9

922 https://d1.almworks.com/.files/structure-1.4.0.jar
923 https://d1.almworks.com/.files/structure-1.4.0.jira43.jar
924 https://d1.almworks.com/.files/structure-1.4.0.jira42.jar
925 https://d1.almworks.com/.files/structure-1.3.2.jar
926 https://d1.almworks.com/.files/structure-1.3.2.jira43.jar
927 https://d1.almworks.com/.files/structure-1.3.2.jira42.jar
928 https://d1.almworks.com/.files/structure-1.3.1.jar
929 https://d1.almworks.com/.files/structure-1.3.1.jira43.jar
930 https://d1.almworks.com/.files/structure-1.3.1.jira42.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
1.3	JIRA 4.4	structure-1.3.0.jar[931]	935	2011-10-24	2012-10-24	c0cc1db5ba33648ee29b66e5f3585bab
1.3	JIRA 4.3	structure-1.3.0.jira43.jar[932]	932	2011-10-24	2012-10-24	79ecd34177d55ce4c01456ec5216ac3f
1.3	JIRA 4.2	structure-1.3.0.jira42.jar[933]	927	2011-10-24	2012-10-24	13430ccee213530ee8bf0e98c24f80dc
1.2	JIRA 4.4	structure-1.2.0.jar[934]	761	2011-08-30	2012-08-30	9be530ae1f9e68b61afdc936ab4bfc50
1.2	JIRA 4.3	structure_jira4.3backport-1.2.0.jar[935]	760	2011-08-30	2012-08-30	9a3a31a0c202f80c9d05f07cb5a08882
1.2	JIRA 4.2	structure_jira4.2backport-1.2.0.jar[936]	759	2011-08-30	2012-08-30	7f9bc6f83028adcb83c2eb436723c33a
1.1.1	JIRA 4.4	structure-1.1.1.jar[937]	661	2011-08-08	2012-08-08	6d0a9245a4c1c6de92eb076295ba7393
1.1.1	JIRA 4.3	structure_jira4.3backport-1.1.1.jar[938]	660	2011-08-08	2012-08-08	22252d97bef859fb7f442e0c9199089f

931 https://d1.almworks.com/.files/structure-1.3.0.jar
932 https://d1.almworks.com/.files/structure-1.3.0.jira43.jar
933 https://d1.almworks.com/.files/structure-1.3.0.jira42.jar
934 https://d1.almworks.com/.files/structure-1.2.0.jar
935 https://d1.almworks.com/.files/structure_jira4.3backport-1.2.0.jar
936 https://d1.almworks.com/.files/structure_jira4.2backport-1.2.0.jar
937 https://d1.almworks.com/.files/structure-1.1.1.jar
938 https://d1.almworks.com/.files/structure_jira4.3backport-1.1.1.jar

Structure Version	Jira version	Download link	Build	Publish Date	End of Life	MD5
1.1.1	JIRA 4.2	structure_jira4.2backport-1.1.1.jar[939]	659	2011-08-08	2012-08-08	f6dc10634dd0a4d02029018ddf3c7df6
1.1	JIRA 4.3	structure-1.1.jar[940]	638	2011-07-29	2012-07-29	7c5d17b69e6d903206936448651a1c753
1.1	JIRA 4.2	structure_jira4.2backport-1.1.jar[941]	637	2011-07-29	2012-07-29	959a9779540428f9325464a4e45714cc
1.1	JIRA 4.1	structure_jira4.1backport-1.1.jar[942]	636	2011-07-29	2012-07-29	b06ac14cafb69df6f6394b8e07d5ec56
1.1	JIRA 4.4 Preview	structure_jira4.4preview-1.1.jar[943]	639	2011-07-29	2011-10-29	c6796a4f3b2392c61b660d03311ebde1
1.0	JIRA 4.3	structure-1.0.jar[944]	572	2011-07-13	2012-07-13	
1.0	JIRA 4.2	structure_jira4.2backport-1.0.jar[945]	571	2011-07-13	2012-07-13	
1.0	JIRA 4.1	structure_jira4.1backport-1.0.jar[946]	570	2011-07-13	2012-07-13	

939 https://d1.almworks.com/.files/structure_jira4.2backport-1.1.1.jar
940 https://d1.almworks.com/.files/structure-1.1.jar
941 https://d1.almworks.com/.files/structure_jira4.2backport-1.1.jar
942 https://d1.almworks.com/.files/structure_jira4.1backport-1.1.jar
943 https://d1.almworks.com/.files/structure_jira4.4preview-1.1.jar
944 https://d1.almworks.com/.files/structure-1.0.jar
945 https://d1.almworks.com/.files/structure_jira4.2backport-1.0.jar
946 https://d1.almworks.com/.files/structure_jira4.1backport-1.0.jar

End of Life:

- For beta versions: Beta versions are scheduled to expire after certain amount of time. You will need to upgrade to another Beta or Production version before that date.
- For other versions: You will be able to use this version of the plugin indefinitely, however, after End-of-Life date, the support for the version is limited.

10 Structure Roadmap

Updated On	09 December 2020
Next Update	Jan 2021

In this roadmap, we'd like to share some of the features the Structure team is going to work on in the near to mid term. The scope of the roadmap is 1 to 2 years.

A few notes and disclaimers about the roadmap:

- We only list **new, important functionality** – we are also going to work on other things, such as improving existing features, improving quality, improving user interface, adding minor features.
- This document lists only upcoming features in Structure. We're also working on Structure extensions, such as Structure.Gantt, which have their own roadmaps.
- The roadmap is subject to change. We will update it periodically so it reflects our current vision.
- We only describe features briefly. If you are interested in the details of some specific feature or can provide feedback and ideas, please let us know at info@almworks.com.[947]

> ⚠ It is our general approach at ALM Works to focus on the quality of the product. Sometimes this means delivering a product later or changing plans and priorities, as unexpected dependencies and challenges appear. Therefore, while we try to adhere to the announced roadmap, by no means should it be considered an obligation from ALM Works, and it should not be relied upon when making purchasing decisions.
>
> If you have any questions about our roadmap policy, please write to info@almworks.com[948].

10.1 Versions and Dates

We generally aim to release a minor version of Structure every 1-2 months and a major version every year. The following is an approximate release schedule for the scope of this roadmap.

Target Month	Jan'21	Mar'21
Version	6.5	7.0

10.2 Roadmap

Here's the list of major features that we're planning to work on next year:

- **Ability to update Jira data based on Structure data** – this will allow users to write certain Structure data into Jira custom fields, links, etc. For example, push Formula results into Jira fields, or update parent status based on children statuses. ✅ Structure 6.0
- **Formula UI** - improvements for formula editor, so it will be easier to type bigger formulas ✅ Structure 6.2
- **Project Category support** - add the possibility to group by project category and use it in formulas ✅ Structure 6.3

947 mailto:info@almworks.com.
948 mailto:info@almworks.com

- **Use non-issue attributes in Structure** - show data such as Fix Version dates or sprint goal inside the structure and use it in formulas ✅ Structure 6.4
- **language improvements -** additional functions for working with dates and supporting arrays.
- **Formula library** – ability to create, share and re-use formulas more easily and even between structures.
- **Structure Templates** – need to create similar structures, but for different projects or teams? Now you will be able to describe the structure in text format, use it as a template for creating new structures, or share it as a file with others.
- **Context Help and Guidance System** – for helping users quickly find a solution within the app.

11 Send Feedback

11.1 Please tell us what you think!

If you have any questions, feature requests or anything else to say about Structure plugin – please tell us:

- Feature and Wish List on UserVoice[949]
- Discussions on Google Group[950]
- Anything at all to structure@almworks.com[951]

949 http://structure.uservoice.com/
950 http://groups.google.com/group/structure-plugin-for-jira
951 mailto:structure@almworks.com